Spatial Multidimensional Sequence Clustering

Ira Assent, Ralph Krieger, Boris Glavic, Thomas Seidl
Data Management and Exploration Group
RWTH Aachen University, Germany
{assent,krieger,glavic,seidl}@cs.rwth-aachen.de

Abstract

Measurements at different time points and positions in large temporal or spatial databases requires effective and efficient data mining techniques. For several parallel measurements, finding clusters of arbitrary length and number of attributes, poses additional challenges. We present a novel algorithm capable of finding parallel clusters in different structural quality parameter values for river sequences used by hydrologists to develop measures for river quality improvements.

1. Introduction

Environmental sensors produce data streams at successive time points, weather stations, observatories and seismographic stations archive similar temporal sequences. Likewise, geosensors generate huge amounts of parallel spatial sequences. In a current project of the European Union on renaturation of rivers, the structural quality of river segments is analyzed. For a spatial database of German rivers, about 120.000 one-hundred-meter segments were evaluated according to 19 different structural criteria (e.g. quality of the riverbed) [14]. They were mapped to quality categories, where a value of "one" indicates perfect quality, while a value of "seven" indicates most severe damages. Figure 1 illustrates a sample river indicating 8 of the 19 attributes. The sequence order of the segments is given by the flowing direction of the rivers.

As the project aims at a quality improvement over the next decades, packages of measures have been suggested for different structural damages. They have been formalized in rules specifying the attribute value constraints and the attributes influenced positively by execution of the respective measure. An example constraint might be that a certain segment has good quality (categories one to three) riverbed and riverbanks and poor river bending (categories five to seven). This could be improved by measures like adding deadwood to positively influence river bending.

Finding and analyzing these patterns helps hydrologists summarize the overall state of rivers, give compact representations of typical situations and review the extent to which these situations are covered by measures envisioned. They can identify those patterns which are problematic, i.e. have low quality ratings, but are not yet covered by measures. In a follow-up step, these measures are annotated by time and cost information to generate an overview over the state of rivers as it might be in the near future if the measures suggested are put into action.

From a computer science point of view, finding the intrinsic structure of these multidimensional sequences is a two-fold task: detect frequent patterns within sequences for all possible subsequence lengths (note that we cannot know the pattern length a priori), then detect parallel occurrences of these patterns.

Patterns are ranges of values (which correspond to several categories of river quality structure) found in several (sub-)sequences. Pattern analysis has to take into account that the data is subjective and fuzzy, because structural quality of rivers was mapped by different individuals. We propose using kernel densities in a density-based clustering approach instead of sequence counting approaches like motif discovery. This effectively detects fuzzy multidimensional sequence patterns. These patterns are clustered efficiently for arbitrary lengths using monotonicity properties. We transform these sequence pattern clusters into a cluster position space such that mining parallel patterns can be reduced to efficient FP-tree frequent itemset mining.

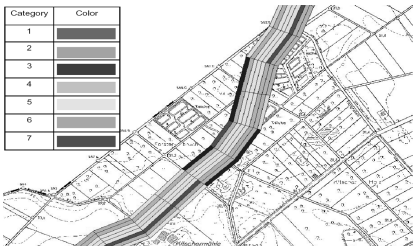

Figure 1. River mapping for eight attributes

2. Subsequence Clustering

Patterns or clusters discovered in sequences help domain experts analyze typical scenarios in the data. Single sequence patterns, however, do not permit direct mapping of possible measures to clusters. As mentioned in the introduction, river measures may affect several structural properties, e.g. the river bank on the left and the right as well as the river bending. Moreover, constraints are often formulated for several attributes as well. In Figure 2 we illustrate two rivers $\mathbf{S}$ and $\mathbf{T}$ with two attributes (riverbed and riverbank). These rivers contain a multicluster MC which consists of two sequence clusters, one cluster C_1 with values $\{\langle 1,2\rangle, \langle 1,3\rangle\}$ in the riverbed attribute and one cluster $C_2 = \{\langle 3,4\rangle\}$ in the bank attribute. The example multicluster is based on the following definition:

Definition 1 ***Multicluster****:*
A set of sequence clusters $\mathbf{C_1}, \ldots, \mathbf{C_n}$ *of length* k *from* n *different attributes is a multicluster* $\mathbf{MC}$ *iff:*

(i) $\mathbf{C_1}, \ldots, \mathbf{C_n}$ *are* ***parallel*** *at position* i*, i.e.*
$\forall j \in \{1 \ldots n\}$ *a pattern* $P_j \in \mathbf{C_j}$ *occurs at position* i

(ii) $\mathbf{C_1} \ldots \mathbf{C_n}$ *occur* ***frequently*** *together, i.e.*
$|\{\mathrm{i} \in \mathbb{N},\ \mathbf{C_1}, \ldots, \mathbf{C_n}$ *are parallel at position* $\mathrm{i}\}| \geq \phi$.

We are thus looking for those positions showing a pattern in each of the sequence clusters (parallel) with a frequency count above the threshold. Sequence clusters consist of dense patterns, i.e. similar patterns occur often.

Definition 2 ***Sequence pattern:***

- *A tupel* $\mathbf{S} = (s_1, \ldots, s_n)$ *of* n *subsequent values at positions* 1 *through* n *is called a* ***sequence*** *of length* n.
- *A database* $\mathbf{DB}$ *is a set of sequences* $\{S^1, ..., S^m\}$.
- *We denote a* ***subsequence*** *of* $\mathbf{S}$ *from position* i *to* j *by* $\mathbf{S}[i,j] = (s_i, \ldots, s_j)$.
- *Whenever we are not interested in the concrete positions of a sequence, but merely in its values, we call this a* ***pattern*** $\mathbf{P} = \langle p_1, \ldots, p_k\rangle$.
- *A pattern* $\mathbf{P}$ *occurs in a database if there is a sequence* $S \in \mathbf{DB}$ *and a position* $i \in \mathbb{N}$ *with* $\mathbf{P} = \mathbf{S}[i, i+k\text{-}1]$.
- *The* ***support*** *of a pattern* $\mathbf{P}$ *is the number of its occurrences in the sequences of the database* $\mathbf{DB}$*:*
$sup(\mathbf{P}) =$
$|\{i \in \mathbb{N}$ *and* $\mathbf{S} \in \mathbf{DB},$ *where* $\mathbf{P} = \mathbf{S}[i, i+k\text{-}1]\}|$

When searching for prevailing patterns, it is important to notice that merely counting of sequence patterns is not sufficient in many scenarios. It is crucial to account for two factors: first, mapping of river structures may be blurred by people's subjective decisions on unclear category boundaries. Second, measures and their constraints may be applicable over several categories and cannot always be fitted exactly to these categorical boundaries. Moreover, small deviations in few segments may be tolerable for the application of a certain measure if this results in longer river courses treatable by a single measure. Hydrologists are thus interested in including "similar" sequences in frequency notions.

Density-based clustering tries to separate dense areas ("clusters") from sparse ones ("noise"). The density of a pattern is determined by evaluating its neighborhood according to some distance function. In our approach, any L_p-Norm can be used, yet the Manhattan norm (L_1) has shown to work well. The L_1 norm reflects the intuition that for each sequence element, the difference to its counterpart should be determined and summed up. We use a weighting function $\mathbf{W}$ to ensure that with greater distance to the pattern evaluated, the influence of neighbors decreases. Any series of monotonous falling values can be used as a weighting function, since distances between nominal sequences are always discrete. All kernel-estimators known from statistics [10] are constantly falling functions. Experiments have shown that weighting functions based on Gaussian kernels ($W_\sigma^{\mathbf{S}}(\mathbf{T_i}) = \exp\left(-d(\mathbf{S}, \mathbf{T_i})^2/2\sigma^2\right)$) perform well in many applications. Using weighting functions based on kernel estimators provides us with a natural way of defining the set of similar sequences to be included in the density evaluation. Whenever the weights assigned drop below a certain significance threshold τ, these sequences should not be considered in the density estimation.

Definition 3 ***Neighborhood and Density****:*

- *The* τ***-neighborhood*** *of a pattern* $\mathbf{P}$*,* $N_\tau(\mathbf{P})$*, is defined as the set of all patterns* $\mathbf{Q}$ *which at least have the influence* τ *on the pattern* $\mathbf{P}$ *evaluated:*
$N_\tau(\mathbf{P}) = \{\mathbf{Q}, where\ W_\sigma^{\mathbf{P}}(\mathbf{Q}) \geq \tau\}$.
- *The* $\mathbf{density}$ *of* $\mathbf{P}$ *is the sum of all kernels in the neighborhood*

$$\mathsf{density}(\mathbf{P}) = \sum_{\mathbf{Q} \in N_\tau(\mathbf{P})} W_\sigma^{\mathbf{P}}(\mathbf{Q}) * sup(\mathbf{Q})$$

- $\mathbf{P}$ *is* ***dense*** *with respect to a density threshold* δ
iff $\mathsf{density}(\mathbf{P}) \geq \delta$.

We are now ready to formulate our cluster notion. As mentioned before, clusters should consist of similar, dense sequences of the same length.

Definition 4 ***Cluster****:*
$\mathbf{C} = \{\mathbf{P}_1, \ldots \mathbf{P}_m\}$ *is a cluster of length* k *with respect to a density threshold* δ *and a compactness parameter* γ *iff:*

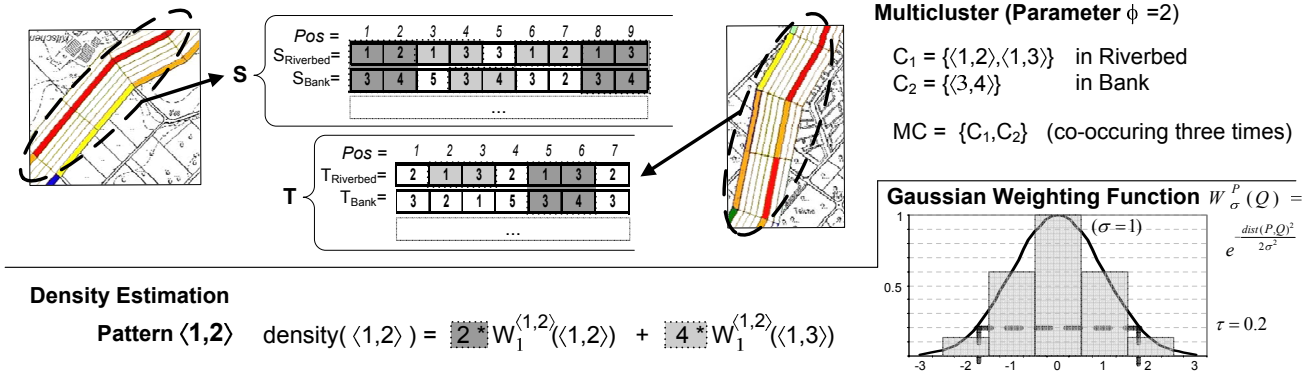

Figure 2. Example multicluster for the river data base

(i) for all patterns $\mathbf{Q}$ *of length* k *not in* $\mathbf{C}$*:* $\mathbf{C} \cup \{\mathbf{Q}\}$ *is not a cluster.* ***(Maximality)***

(ii) for all patterns $\mathbf{P}_i$: $\mathsf{density}(\mathbf{P}_i) \geq \delta$. ***(Density)***

(iii) between any pair of patterns $\mathbf{P}_i, \mathbf{P}_j \in \mathbf{C}$ *there is a chain of patterns* $(\mathbf{Q}_1, \ldots, \mathbf{Q}_n) \in \mathbf{C}$ *such that* $\mathsf{dist}(\mathbf{P}_i, \mathbf{Q}_1) \leq \gamma$, $\mathsf{dist}(\mathbf{Q}_k, \mathbf{Q}_{k+1}) \leq \gamma$ *and* $\mathsf{dist}(\mathbf{Q}_n, \mathbf{P}_j) \leq \gamma$. ***(Compactness)***

Put informally, we are thus looking for as large clusters as possible (maximality), where elements within clusters are all dense (density) and at most γ apart from each other (compactness).

Figure 2 illustrates the definition of density and clustering. The upper part visualizes two sequences $\mathbf{S}$ and $\mathbf{T}$ with two exemplary attributes, the riverbed and the bank. In the lower part of the figure the density-value for a pattern $\langle 1, 2\rangle$ is calculated. We assume a significance threshold of $\tau = 0.1$. The Gaussian weighting function drops below $\tau = 0.2$ for sequences having a distance higher than 1.8 from the point evaluated. Thus in our ordinal setting only sequences with a distances of or less 1 have significant influence: $\exp(-1^2/2) \approx 0.6 > \tau$ and $\exp(-2^2/2) \approx 0.13 < \tau$. In our example the τ-neighborhood of pattern $\langle 1, 2\rangle$ contains the sequence $\langle 1, 2\rangle$ itself starting at two positions S_1 and S_6 (distance zero) and $\langle 1, 3\rangle$ starting at four positions S_3, S_8 and T_2, T_5 (distance one). Thus the density-value for $\langle 1, 2\rangle$ is $2 * W_\sigma^{\langle 1,2\rangle}(\langle 1, 2\rangle) + 4 * W_\sigma^{\langle 1,2\rangle}(\langle 1, 3\rangle) = 2 * 1 + 4 * 0.6 = 4.4$. For a density-threshold of e.g. $\delta = 3$ the pattern $\langle 1, 2\rangle$ is considered dense.

The definition of multiclusters is based on sequence clusters of arbitrary lengths. To detect clusters of arbitrary length, a naive approach might be to simply re-run a density-based subspace clustering algorithm for each length value to detect all possible clusterings. Obviously, this leaves room for efficiency improvement. We therefore use a monotonicity property between clusters of different lengths which can be used to speed up clustering.

Theorem 1 ***Density Monotonicity****:*
For any two patterns $\mathbf{P}, \mathbf{Q}$ *of length* k *and their respective prefix/suffix* $\mathbf{P}', \mathbf{Q}'$ *of length* $k-1$ *holds:*

(1) $\mathbf{Q} \in N_\epsilon(\mathbf{P}) \Rightarrow \mathbf{Q}' \in N_\epsilon(\mathbf{P}')$

(2) $\mathsf{density}(\mathbf{P}) \leq \mathsf{density}(\mathbf{P}')$

Since density and neighborship are monotone, we can conclude that clusters are monotone as well:

Theorem 2 ***Cluster Monotonicity****:*
For any cluster $\mathbf{C}$ *of length* k*, there is a cluster* $\mathbf{C}'$ *of length* $k-1$ *such that for any pattern* $\mathbf{P}$ *and its prefix/suffix* $\mathbf{P}'$ *holds:* $\mathbf{P} \in \mathbf{C} \Rightarrow \mathbf{P}' \in \mathbf{C}'$

Proofs are given in an extended version of this paper. To speed up the calculation of multiclusters of arbitrary length two kinds of monotonicity properties can be utilized (see Section 2). The proposed algorithm uses two steps to make use of each monotonicity property. We then model parallelism between these clusters and give a transformation which translates this problem into an efficiently solvable frequent itemset mining task.

3. Multicluster Algorithm MC

For the algorithm, it is most important to efficiently calculate the density value of patterns. Thus, it is crucial to quickly retrieve the neighborhood of a subsequence. To identify clusters of arbitrary length the MC algorithm bottom-up identifies successively longer clusters. Existing index structures which support neighborhood queries cannot handle arbitrary lengths. A novel index (Fig. 3) is constructed by scanning once over each sequence. Each pattern of a fixed starting length is added. A pattern is represented by a path of labeled nodes from the root to a leaf. To later determine the density value of a pattern the support is annotated at corresponding leaf nodes. Further on the index

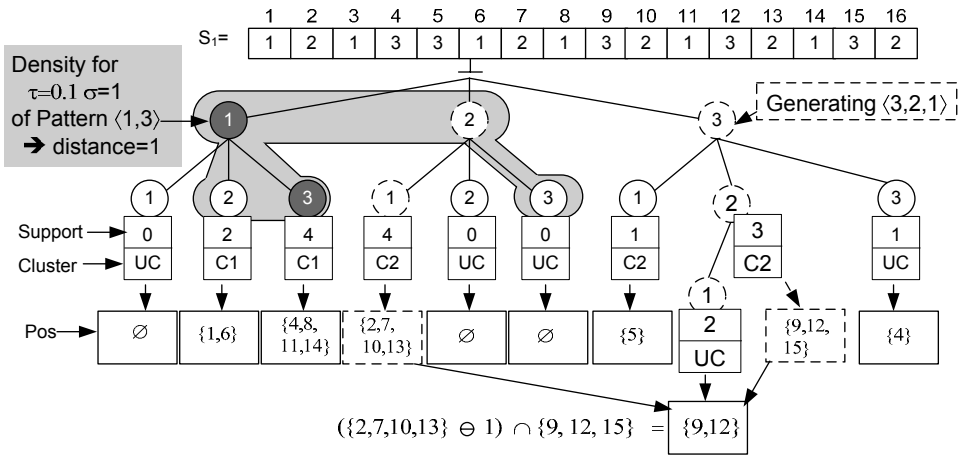

Figure 3. Hierarchical index structure

structure stores the position list (all starting points) for each pattern stored on hard disk for scalability reasons. A cluster identifier (e.g. $C1$) or the flag *unclassified* (UC) is stored at each leaf node. The MC algorithm uses this flag to efficiently calculate the transitive closure (compactness in Def. 4) for a dense pattern. The index efficiently supports neighborhood queries and density calculations by selecting the appropriate node ranges while descending the tree and summing up weighted support values.

The index structure must be able to calculate the position list and support for constantly growing patterns. Therefore, the position list for a pattern $\mathbf{P}$ of length k can be calculated by using its $k-1$ prefix and suffix. Intuitively, the pattern $\mathbf{P}$ can only occur in a sequence if its $k-1$ prefix also starts at $\mathbf{P}$s starting position and its $k-1$ suffix ends where $\mathbf{P}$ ends. This means that no extra database scans are necessary to determine its occurrence - we simply take a look at all the starting positions of its prefix and determine its intersection with its suffix shifted by one. Since both are shorter by one, they must have been processed earlier. In Fig. 3 pattern (3,2,1) is generated from (3,2) prefix and the (2,1) suffix shifted by one position.

An overview of the MC algorithm is presented in Figure 4. For each length, the index structure is queried for all cluster candidates. Looping over all candidates, each sequence is checked for density, and if applicable, expanded to a cluster. The algorithm to calculate the corresponding cluster candidates is based on the discovered clusters of the previous step. Looping over all patterns of all clusters, it tries to extend each pattern. For this purpose the suffix of $length-1$ is extracted from each pattern and all patterns which start with the extracted suffix are queried from the index. Each queried pattern which satisfies the density property is then used to create an elongated pattern by concatenating the appropriate prefix and suffix. These queries are also supported by our index structure. The elongated patterns are finally assigned to a new cluster candidate.

Having discovered dense patterns and combined them to clusters, we identify those clusters which are parallel in the dataset. Since for any cluster of length k all corresponding clusters of length $k-1$ have previously been detected, we use the monotonicity property presented in Lemma 2. An important property of multiclusters is that a subsequence of a specific length may belong to no more than one cluster. Hence any position in a sequence is the starting point for at most one density based cluster of a fixed length. This property is used to transform the sequence database from a value representation to a cluster representation. After this transformation it is possible to discover parallel clusters by efficient frequent itemset mining techniques.

We use the following representation to apply the frequent itemset mining: clusters starting at the same position in different attributes are combined to one itemset. The clusters are identified by their cluster-ids. A frequent itemset contains often occurring multiclusters in which each cluster belongs to a different attribute.

We use the Frequent Pattern (FP) growth algorithm proposed by Han et al. for the extraction of frequent itemsets [9]. The tree representation is very suitable for our task

```
MC-Clustering(db,length_start,length_max)
foreach length from length_start to length_max
   foreach clustCand in candidateSet
      foreach seq in clustCand do
         if isDense(seq, index)
            ExpandToCluster(seq, clustCand);
   index.prune(length-1);
   CreateClusterCandSet(length+1);
```

Figure 4. Subsequence clustering algorithm

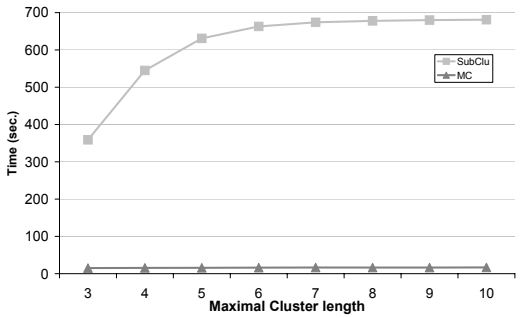

Figure 5. MC and $SUBCLU$ on synthetic data

since database scans are avoided. Thus the result of the FP-tree algorithm contains multiclusters as described in Def. 1. In order to find multiclusters of any length the FP-tree algorithm is started for all different lengths for which clusters have been found. Since the FP-tree works extremely fast on the database of clusters, the loop over all different lengths is efficient.

4. Experimental Evaluation

As mentioned in Section 2, a naive approach for detecting clusters of arbitrary length would be to re-run a density-based subspace clustering algorithm like $SUBCLU$ [11] for all possible lengths. Since $SUBCLU$ was not developed to cluster subsequences we had to extend the original implementation by our density notion. Figure 5 illustrates the runtime of both algorithms on synthetic data. Note that MC is faster by an order of magnitude. The runtime of both algorithms depends on the number of subsequences belonging to a cluster. Since longer subsequences are less often dense the time to investigate longer subspace subsequence is nearly constant. As far as the quality of the result is concerned, both algorithms discovered the main patterns of the six subspace subsequence clusters hidden in the database.

For the river dataset a value for σ between 0.7 and 0.85 has shaped up as a reasonable parameter. The MC algorithm identifies more than a thousand clusters of length four by using a value for σ of 0.7. Many of those clusters do not show when mining clusters of length five. By using a higher value for σ more subsequences are considered as similar and cluster count drops between lengths five and six. For this dataset experiments have shown that independent of the σ value, parallel patterns in many attributes can be found only at length four to six. Beyond this length, parallel patterns are rare. This an interesting result for the hydrologists studying this data. They find that they should develop packages of measures in this range and estimate costs for about 500 meters river improvement. Figure 6 illustrates the runtime of the MC algorithm for phase one (searching for clusters of arbitrary length) and for phase two (searching for parallel clusters) separately as well as for both. As we can see, the time requirements for mining all clusters of arbitrary length are distributed rather evenly between the two phases. The total time for mining multiclusters of arbitrary pattern length demonstrates the efficiency of our approach. Note that with increasing maximal length, few additional dense patterns are detected such that the increase in time consumption slows down. The steepest ascent is for lengths of up to six, which corresponds to our result findings.

Similar to the synthetic dataset, we also applied $SUBCLU$ on this real world dataset. However, even the first iteration of $SUBCLU$ for subspace subsequence clusters of length ten did not finished after ten hours. One reason why MC works extremely faster on the investigated dataset than $SUBCLU$ is the efficient combination of dimensions using an FP-tree as done by MC. Another reason are the time consuming neighborhood queries on the nineteen-dimensional data points. Even the use of index structures like the R-Tree does not speed up these neighborhood queries in these high dimensionalities.

For hydrologists to see how the detected multiclusters are distributed and check in which areas the designed measures are applicable or where additional measure engineering is required, we visualize multiclusters in a geographical information system. Additional tools for hydrologists give detailed information beyond this summary. Experts may browse clusters for an overview of the attributes contained in clusters, their value distributions as well as their occurrence in the rivers. Moreover, individual attributes and river values may be checked for containment in clusters. By joining this information with the packages of measures designed, field experts can easily identify areas which are not yet met by measures. Annotating the measures with cost and duration information, political decision making is supported. Hydrologists used the information derived from these clusters to build a decision support system [3] that gives concise summaries as well as detailed inspection of

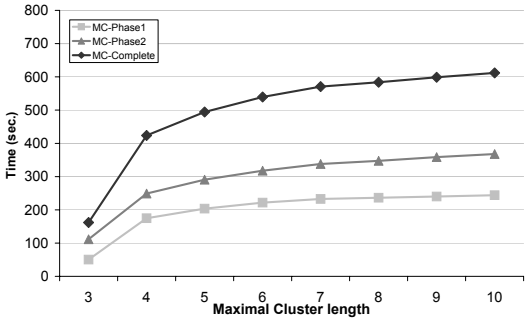

Figure 6. Runtime on real world data

the state of the rivers as it is now as well as a prognosis for future development depending on the packages of measures chosen.

Our algorithm is useful for other multidimensional sequence data applications. We evaluated weather station data from the "National Fire Danger Rating Retrieval System" from 1998 to 2005 about temperature, relative humidity, wind speed, etc. [7]. The data includes indices like the "Ignition Component", i.e. the probability that a firebrand, if present, would start a fire. These indices are influenced by many variables forming natural multidimensional patterns. MC detects this correlation between the index variables and all measured sensor data. Thus the MC algorithm indeed captures the data's intrinsic structure.

5. Related Work

The analysis of sequence data has recently gained a lot of attention. Recordings of data at successive time points have been studied in time series mining (e.g. [6, 13]). Most of these approaches, however, aim at finding patterns of values which do not have to directly follow one another, but may have other values in-between (as in sequential frequent itemset mining [1, 2]). A recent approach, dubbed motif mining, is similar to our approach in that it searches those patterns which have the highest count of similar subsequences [16]. Matching within some range is used to determine the frequency. However, neighbors are not weighted. Moreover, parallel patterns are not discussed since the application targeted is one-dimensional time series. While noise is removed in motif discovery as well, we found density-based clustering to be more useful in handling the fuzzy and blurred river data collected by several individuals. Numerous clustering methods have been proposed in the literature, including partitioning clustering, e.g. the well-known k-means algorithm [15]. These algorithms require the specification of the number of clusters to be found and can only detect convex cluster regions. Categorical clustering methods work well for categorical data where the notion of neighborhood is not meaningful [8, 17]. Density-based algorithms use a function to determine the density of the neighborhood of each point and use a connectivity-notion to assign similar points to the same cluster [5, 10]. Density-based clustering is robust to noise since it clusters only those points or sequences above some noise threshold as discussed in [4]. Moreover, it naturally incorporates neighboring objects into its cluster definition.

6. Conclusion and Further work

Domain experts are supported in their need for pattern detection in sequence databases such as river quality records. The information derived using our approach is incorporated in a decision support system devised by hydrologists. Future work will concentrate on integrating GIS information to handle non-sequence spatial information as well and to extend the model to graph structures [12].

Acknowledgments: We would like to thank K. Kailing, H.-P. Kriegel and P. Kroeger for providing the original source code of SUBCLU.

References

[1] Agrawal, R., Srikant, R. Mining sequential patterns. In *ICDE*, pages 3–14, 1995.

[2] Ayres, J. et al. Sequential pattern mining using a bitmap representation. In *KDD*, pages 429–435, 2002.

[3] Bartusseck, S. Regelbasiertes Entscheidungsunterstützungssystem (DSS) zur Bewertung von Maßnahmenplänen gemäß EG-WRRL. *Forum für Hydrologie und Wasserbewirtschaftung*, 10, 2005.

[4] Denton, A. Density-based clustering of time series subsequences. In *ICDM*, 2004.

[5] Ester, M. et al. A density-based algorithm for discovering clusters in large spatial databases. In *KDD*, pages 226–231, 1996.

[6] Faloutsos, C., Ranganathan, M., Manolopoulos, Y. Fast subsequence matching in time-series databases. In *SIGMOD*, pages 419–429, 1994.

[7] Georgia Forestry Commission. Weather data retrieval. *http://weather.gfc.state.ga.us*, 2005.

[8] Guha, S., Rastogi, R., Shim, K. A robust clustering algorithm for categorical attributes. In *ICDE*, pages 512–521, 1999.

[9] Han, J., Pei, J., Yin, Y. Mining frequent patterns without candidate generation. In *SIGMOD*, pages 1–12, 2000.

[10] Hinneburg, A., Keim, D. An efficient approach to clustering in large multimedia databases with noise. In *KDD*, pages 58–65, 1998.

[11] Kailing, K., Kriegel, H.-P., Kroeger, P. Density-connected subspace clustering for high-dimensional data. In *ICDM*, pages 246–257, 2004.

[12] Kailing K., Kriegel H.-P., Schnauer S., Seidl T. Efficient similarity search for hierarchical data in large databases. In *EDBT*, pages 676–693, 2004.

[13] Keogh, E., Chakrabarti, K., Mehrotra, S., Pazzani, M. Locally adaptive dimensionality reduction for indexing large time series databases. In *SIGMOD*, pages 151–162, 2001.

[14] LUA NRW (North Rhine-Westphalia State Environment Agency). River quality data. http://www.lua.nrw.de, 2003.

[15] MacQueen, J. Some methods for classification and analysis of multivariate observations. In *Berkeley Symp. Math. stat. & prob.*, pages 281–297, 1967.

[16] Patel, P. Keogh, E., Lin, J., Lonardi, S. Mining motifs in massive time series databases. In *ICDM*, 2002.

[17] Zaki M., Peters M., Assent I., Seidl T. Clicks: An effective algorithm for mining subspace clusters in categorical datasets. In *SIGKDD*, pages 355–356, 2005.