Semigeometric Tiling of Event Sequences

Andreas Henelius[1(✉)], Isak Karlsson[2], Panagiotis Papapetrou[2], Antti Ukkonen[1], and Kai Puolamäki[1]

[1] Finnish Institute of Occupational Health, PO Box 40, 00251 Helsinki, Finland
{andreas.henelius,antti.ukkonen,kai.puolamaki}@ttl.fi
[2] Department of Computer and Systems Sciences, Stockholm University, Forum 100, 164 40 Kista, Sweden
{isak-kar,panagiotis}@dsv.su.se

Abstract. Event sequences are ubiquitous, e.g., in finance, medicine, and social media. Often the same underlying phenomenon, such as television advertisements during Superbowl, is reflected in independent event sequences, like different Twitter users. It is hence of interest to find combinations of temporal segments and subsets of sequences where an event of interest, like a particular hashtag, has an increased occurrence probability. Such patterns allow exploration of the event sequences in terms of their evolving temporal dynamics, and provide more fine-grained insights to the data than what for example straightforward clustering can reveal. We formulate the task of finding such patterns as a novel matrix tiling problem, and propose two algorithms for solving it. Our first algorithm is a greedy set-cover heuristic, while in the second approach we view the problem as time-series segmentation. We apply the algorithms on real and artificial datasets and obtain promising results. The software related to this paper is available at https://github.com/bwrc/semigeom-r.

Keywords: Event sequences · Tiling · Covering · Binary matrices

1 Introduction

Phenomena that evolve over time appear in a wide range of application domains including finance (e.g., stock markets [16]), process monitoring (e.g., telecommunications systems [19]), medicine (e.g., biosignals or electronic patient records, [2]), geoscience (e.g., weather or geological measurements [26]), and mobile sensors [15]. Data from such domains can often be represented as *event sequences*, i.e., sequences of labels that correspond to various events associated with a timestamp of the occurrence of the event. Many processes generate such sequences naturally, or a low level signal can be discretised into an event sequence by applying some suitable method, such as SAX [25].

Given multiple time-aligned event sequences, an important problem is to find similarities between them, allowing the detection of underlying higher-level patterns in the data. This problem has been approached using several different techniques, such as segmentation [3], motif detection [29], and clustering [12,14,31].

P. Frasconi et al. (Eds.): ECML PKDD 2016, Part I, LNAI 9851, pp. 329–344, 2016.
DOI: 10.1007/978-3-319-46128-1_21

Finding similarities becomes more challenging when the event sequences are non-stationary, which is often the case in real application domains, such as volatile stock markets or rapidly changing social media streams. In a collection of non-stationary event sequences, interesting local patterns emerge as subsets of event sequences synchronise and desynchronise over short periods of time. Hence, different event sequences are related to each other during different time periods, forming groupings of intra-related sequences. More importantly, these groupings are not static, but they can also evolve over time.

In this paper, we study the following problem: *given multiple event sequences, identify continuous time segments where subgroups of these sequences exhibit similar behaviour*. This formulation is generic and goes beyond state-of-the-art sequence clustering and segmentation problems, since the objective is to identify subgroups of sequence segments that share dominant local trends. Such subgroups can reveal local temporal similarities and dependencies between the sequences belonging to the same subgroup, which would otherwise be hidden by the global trends and structure of the sequences. Our problem is applicable to several domains. For example, in stock market analysis, we may want to identify subgroups of stocks that exhibit similar trends within different time periods. Identifying such groupings of trends and dependencies can provide insights and reveal potential underlying socio-economic events partly affecting the market.

We approach our problem as a *matrix tiling problem*, where event sequences are compactly represented as a matrix, where each row holds an event sequence and each column corresponds to a time point. Hence, our task now becomes equivalent to finding *tiles* in the matrix. A tile consists of a consecutive range of columns (time points) and an arbitrary set of rows (event sequences) of the input matrix. Unlike tiles that are fully geometric (both rows and columns must be consecutive), or combinatorial (both rows and columns can be chosen arbitrarily), we call our tiles *semigeometric*, since only one dimension (time) must form a contiguous segment. A semigeometric tile, thus, represents a group formed by a subset of event sequences sharing the same dominant feature for the duration of the tile. We illustrate our approach with an example using stock index data.

Example. Figure 1 shows daily closing values of ten stock market indices during 1995–2000. Segments representing patterns of economic decline are shown in (a) while segments of economic growth are shown in (b). The discretisation process used for this dataset is described in detail in Sect. 4. Each of the four panels shows a *tiling* of the stock indices using the `MaxTile` and `GlobalTile` algorithms presented in this paper. Vertically aligned segments with the same colour belong to the same *tile* and represent a region where the event series share the dominant feature; here economic growth or decline. More precisely, the coloured tiles in (a) represent segments of economic decline, whereas in (b) the tiles represent segments with economic growth. Applying the proposed tiling method allows us to discover interesting temporal patterns in the data that can be explained by econo-political events. For instance, all algorithms here detect the concurrent economic decline due to the Russian economic crisis in the autumn of 1998 (and the concurrent rebound later the same year).

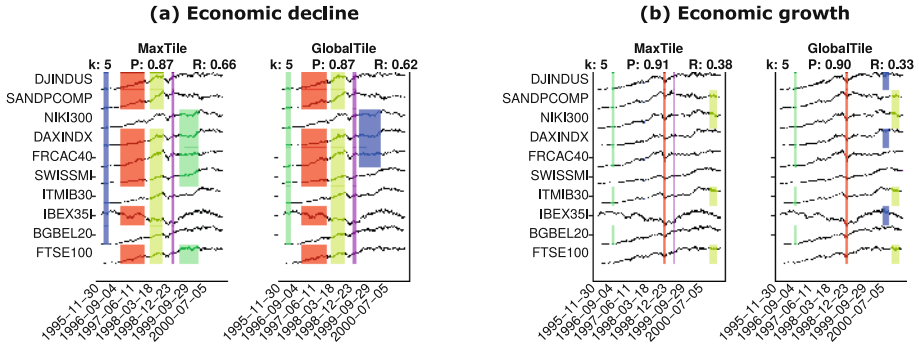


Fig. 1. (a) Periods of economic decline (`stock indices (decline)`) and (b) growth (`stock indices (growth)`) for 10 different stock indices. The coloured regions are identified using the algorithms presented in this paper. The horizontal axis shows days between the years 1995–2000. Vertically aligned segments with the same colour are part of the same tile. Note that the tiles during decline in Figure (a) do not overlap with the tiles during growth in Figure (b). (Color figure online)

Related Work. The problem of finding regions in data matrices with characteristic properties has been extensively studied in multiple contexts; e.g., biclustering (e.g., [5,14]), segmentation (e.g., [12]), tiling (e.g., [10,11,21,33]) and data streams (e.g., [22]). The problem studied in this paper differs from the traditional biclustering and tiling problems in the sense that while we are interested in simultaneously clustering dimensions (rows) over time (columns) based on a given similarity feature of the data, we require the columns in a tile to be consecutive. As discussed below, the temporal ordering has significant impact on the computational efficiency and implementation of the algorithms. In previous work on column-coherent biclusters (see, e.g., [23,27,34]) a specific structure is enforced on the structure column structure of the clusters. The problem of local correlation patterns discussed in [32] is related, but relates to local correlations in time and not precision of tiles. In contrast to these, we define the quality of our tiles in terms of precision and recall, and frame the task of finding a tiling as a covering problem that allows us to build upon existing efficient algorithms.

Contributions. In this paper, we introduce and formulate the novel problem of semigeometric tiling of event sequences and present two algorithms for solving it. The first algorithm, called `MaxTile` is a greedy approach based on the set-cover problem, while the second algorithm, called `GlobalTile`, employs dynamic programming. In addition, we introduce three metrics to assess the quality of a tiling. We also discuss the complexity of the problem and show its connection to two well-known **NP**-hard problems. Finally, we demonstrate the utility of the proposed methods through an extensive empirical evaluation on both real and synthetic datasets.

2 Problem Definition

In this section we introduce the notation used in the paper and formalise the problem we address. We consider an $n \times m$ matrix X with elements from a finite alphabet Σ. The rows of X are event series and the columns correspond to time instances. With X_{ij} we denote the element of X on row i and in column j. For real-valued event series we assume that some suitable discretisation method (e.g., using Fourier coefficients or wavelets [1,4], linear or non-linear piecewise approximations [6,17] or symbolic representations [25]) has been applied in a pre-processing step. Unless otherwise specified, we assume that $\Sigma = \{0, 1\}$, i.e., X is a binary matrix. Generalisation to larger alphabets is rather straightforward and is discussed below, though for simplicity we focus on binary alphabets.

Given a matrix X, our objective is to identify *tiles*. These are consecutive segments in which a subset of the rows of X contain mostly 1s. Formally, a tile t is a three-element tuple (R, a, b) where R is a set of rows of X, and a and b are endpoints of the tile, i.e., column indices corresponding to the beginning and end of the tile, respectively. Below, we use R_t, a_t, and b_t to refer to the tuple elements for a tile t. The *coverage* of the tile t, denoted $C(t)$, is a set of matrix elements belonging to the tile, i.e.,

$$C(t) = \{(i, j) \mid i \in R_t \text{ and } a_t \leq j \leq b_t\}.$$

Tiles t and t' *overlap* unless their intersection $C(t) \cap C(t')$ is empty. Finally, a *tiling* T is a set of possibly overlapping tiles. The coverage of T, denoted $C(T)$, is the union of the covers of each tile in T, and the *weight* $W(T)$ of T is the sum of the elements in $C(T)$, i.e., $C(T) = \bigcup_{t \in T} C(t)$ and $W(T) = \sum_{(i,j) \in C(T)} X_{ij}$. Intuitively, a tiling is good if it (a) has a large weight (covers mainly ones and only a few zeros), (b) covers as many of the ones in X as possible, and (c) uses as few tiles as possible. Requirement (c) is easily achieved by constraining the cardinality of the tiling, while (a) and (b) can be naturally formalised in terms of *precision* and *recall*. Precision is the fraction of ones in the coverage $C(T)$ and recall is the fraction of all ones in X belonging to $C(T)$, i.e.,

$$\text{precision}(T) = \frac{W(T)}{|C(T)|}, \quad \text{and} \quad \text{recall}(T) = \frac{W(T)}{\sum_i \sum_j X_{ij}}.$$

Optimising both precision and recall simultaneously requires formulating a bicriteria optimisation problem, or using some aggregate objective function, such as the F1-measure. Here we opt for maximising recall with a lower bound constraint on precision and an upper bound constraint on the cardinality of the tiling T. This is a natural definition, as it aims to find a tiling that explains as much of the 1s in X as possible without using tiles that are too noisy (precision constraint), nor is too complex (cardinality constraint). Importantly, in most practical situations the constraint on precision is easy for the user to set given that the density of X (i.e., fraction of 1s) is known. It should be noted that the parameters could also be chosen using an approach based on, e.g., the minimum length description (MDL) principle.

We now formulate the main problem studied in this paper:

Problem 1. Semigeometric Tiling Given an $n \times m$ binary matrix X, an integer k, and a real number $\alpha \in [0, 1]$, find the tiling T maximising recall(T) subject to the constraints precision($\{t\}$) $\geq 1 - \alpha$ for each tile $t \in T$ and $|T| \leq k$.

Problem Complexity. Upon first inspection Semigeometric Tiling seems very similar to two well-known problems: the nongeometric tiling problem (i.e., the combinatorial tiling problem) [10,11], where *both rows and columns are unordered*, and to the rectilinear picture compression problem [9, problem SR25], where *both columns and rows have a fixed order*. In the semigeometric case, the smallest tiling covering all 1s in the input matrix can be found in polynomial time.

Theorem 1. *A minimum-cardinality semigeometric tiling for an $n \times m$ binary matrix X with perfect precision and recall can be found in polynomial time.*

The proof follows from known results for the problem of covering a vertically convex polygon with rectangles [8,13,20].

Hence, the problem of semigeometric tiling is computationally different from the nongeometric and fully geometric tiling problems. Notice, however, that Theorem 1 does not imply that Semigeometric Tiling is tractable; it merely suggests that the complexity of this problem is not trivially **NP**-hard. There are also other reasons to argue why this might be the case. Namely, for the nongeometric tiling problem it is easy to establish that merely finding the largest tile is **NP**-hard. In our case this sub-problem is also easy, as we discuss below. However, the actual complexity of Problem 1 is an open question.

Generalisation to Multi-letter Alphabets. It is important to note that generalisation to multi-letter alphabets follows directly from the above given definitions. Multi-letter tilings are constructed by first finding the individual tiling for each letter, after which recall is found by selecting k tiles from the individual tilings, such that recall is maximised. If different precisions are used for different alphabet letters, the minimum precision of the multi-letter tiling corresponds to the smallest single-letter precision.

3 Algorithms

In this section we describe in detail two algorithms (`MaxTile` and `GlobalTile`) for solving the Semigeometric Tiling problem (Problem 1).

3.1 Overview

We approach Problem 1 using two slightly different, yet standard strategies that make use of the property that the columns of the input matrix X are ordered. An example of how our algorithms work on a small example dataset containing three overlapping tiles is shown in Fig. 2.

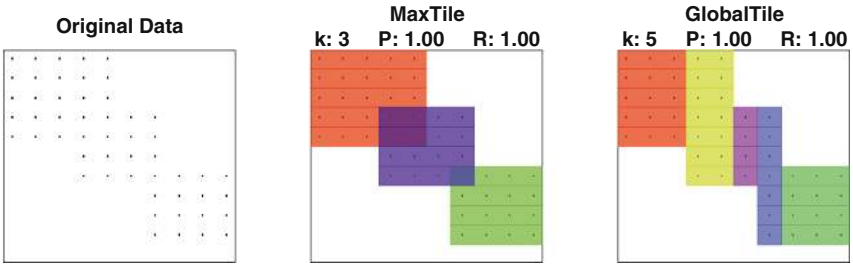


Fig. 2. Example comparing the output between the two algorithms. **k** stands for cardinality, **P** for precision and **R** for recall. The maximum cardinality was set to 5 and the lower bound for precision to 0.95. The solution found by `GlobalTile` (rightmost figure) is also equal to the vertical decomposition (see Sect. 3.3 and Proposition 2) of the original tiling.

Both `MaxTile` and `GlobalTile` algorithms are based on the simple observation that given X and two endpoints a and b, we can easily find the *maximum recall tile* (R, a, b) satisfying the precision constraint:

Definition 1. *Given the $n \times m$ binary matrix X and the precision constraint α, the* maximum recall tile *with endpoints a and b is defined as (R^*, a, b), where R^* is found by maximising recall subject to the precision constraint α.*

We can find the maximum recall tile for every choice of a and b in $O(n \log n)$ time by using cumulative sums of rows to find the number of ones within a row for a given interval and by sorting the rows by the number of ones. Some choices of a and b may result in an empty maximum recall tile, e.g., if there are no rows with enough 1s to satisfy the precision constraint. We omit such tiles, and obtain the set $\mathcal{T}$ of at most $\binom{m}{2}$ candidate tiles.

`MaxTile` and `GlobalTile` differ in the way the tiling is constructed from candidate tiles. `MaxTile` greedily solves a cardinality-constrained set-cover problem, while `GlobalTile` treats the problem as a time-series segmentation task and finds the tiling using dynamic-programming, a side effect of which is that `GlobalTile` cannot produce overlapping tilings. `MaxTile` has no such constraint. This is seen in Fig. 2, where the `MaxTile` tiling is overlapping and the `GlobalTile` tiling consists of vertical non-overlapping stripes.

3.2 MaxTile

The `MaxTile` algorithm is based on a straightforward mapping of the tiling problem to a set cover problem. `MaxTile` first finds all maximum recall tiles (Definition 1) in the data matrix X with a precision of at least $1 - \alpha$. A subset of k, possibly overlapping, tiles maximising recall is then chosen from these tiles. The `MaxTile` algorithm is presented in Algorithm 1 and has a complexity of $\mathcal{O}\left(m^2 n \log n\right)$. The algorithm maintains a tiling T, and always adds a tile $t \notin T$ maximising the marginal gain, i.e., $\text{recall}(T \cup t) - \text{recall}(T)$.

input : $n \times m$ binary matrix X, precision threshold α, max no of tiles k.

Let $\mathcal{T} \leftarrow \cup_{1 \le a \le b \le m} t_{ab}$, where $t_{ab} = (R, a, b)$ is a tile with $R \subseteq [n]$ with the highest recall and precision $\ge \alpha$

Greedily find the tiling $T \subseteq \mathcal{T}$, $\mid T \mid = k$ maximising recall of X.

return The tiling T.

Algorithm 1. `MaxTile`

input : $n \times m$ binary matrix X, precision threshold α, max no of tiles k.

Use dynamic programming to compute all l-segmentations S_l up to $l = 2k + 1$.
$\mathcal{T} \leftarrow \emptyset$
for $l = k$ *to* $2k + 1$ **do**
    $T_l \leftarrow$ all tiles from S_l
    if T_l *has more than* k *tiles* **then**
        $T_l \leftarrow$ the k tiles $t \in T_l$ having highest $W(t)$
    end
    insert T_l into $\mathcal{T}$
end
return *The tiling* $T \in \mathcal{T}$ *that has the highest* $W(T)$

Algorithm 2. `GlobalTile`

Proposition 1. *The function* $\mathrm{recall}(T)$ *is submodular, i.e., when* $T' \subset T$, *and* t *is some tile not in either* T *or* T',

$$\mathrm{recall}(T' \cup t) - \mathrm{recall}(T') \ge \mathrm{recall}(T \cup t) - \mathrm{recall}(T).$$

The proof follows the usual argument for covering functions.

Since $\mathrm{recall}(T)$ is submodular, we can employ the same optimisation as in, e.g., [24]: we maintain a priority queue of known marginal gains for every tile and only recompute the marginal gain for a given tile if it is larger than the best marginal gain observed for the current T.

Also, note that $\mathrm{recall}(\emptyset) = 0$, and that $\mathrm{recall}(T)$ increases monotonically as T grows. This, together with the submodularity of $\mathrm{recall}(T)$ and well-known results from [28], yields that the greedy algorithm has a constant approximation factor of $1 - 1/e$, i.e., the recall of the `MaxTile` tiling is at least $1 - 1/e$ times the optimal recall with k tiles.

3.3 GlobalTile

To design the `GlobalTile` algorithm we view the Semigeometric Tiling problem as a time-series segmentation task, the objective of which is to partition a given time-series into l non-overlapping, contiguous intervals, called *segments*. The partition is found by optimising the sum of segment-wise scores. We call this partition an *l-segmentation*. The input matrix X can be viewed as a multidimensional time-series, where a contiguous range of columns from column a to column b forms a segment. The segment-specific score is defined as the recall of the maximum recall tile (R^*, a, b). The mapping between an l-segmentation and a tiling is thus straightforward. Every segment in an l-segmentation has two endpoints, a and b, given which we can compute the maximum recall tile (R^*, a, b) as specified in Definition 1. Since (R^*, a, b) may be empty for some

choices of a and b, some segments may correspond to empty tiles. Therefore, every l-segmentation maps to a tiling of cardinality *at most* l. In summary, given the matrix X and the integer k, the idea of `GlobalTile` is to (a) compute an optimal $l \geq k$-segmentation of X, and (b) turn this l-segmentation into a tiling of size $k \leq l$.

Importantly, an optimal l-segmentation can be computed in polynomial time with dynamic programming [3]. Unfortunately this does not give a polynomial time algorithm for Problem 1, because the parameters l and k do not match one-to-one, and the resulting segmentation must be non-overlapping. This constraint is not present in Problem 1. However, we can still use an l-segmentation as a building block in our algorithm as follows. Given the input X and the integer k, the `GlobalTile` algorithm first finds all l-segmentations of X from $l = k$ up to $l = 2k + 1$. Let T_l denote the tiling corresponding to a given l-segmentation. Since T_l may contain up to $l \leq 2k + 1$ tiles, the algorithm keeps only the k largest tiles of every T_l to maximise recall. This pruning step is repeated for every tiling T_l with $l \in \{k, \ldots, 2k + 1\}$, and the algorithm returns the tiling T_l having the highest recall. Pseudocode of `GlobalTile` is shown in Algorithm 2.

Next we discuss some properties of the `GlobalTile` algorithm. In the following we call a tiling T of cardinality k a *k-tiling*. In any k-tiling, tiles (R_1, a_1, b_1) and (R_2, a_2, b_2) are *vertically non-overlapping* if $b_1 < a_2$ and $b_2 < a_1$. A tiling is vertically non-overlapping if all of its tiles are non-overlapping. The *vertical decomposition* of T is the vertically non-overlapping tiling T' with smallest cardinality that has *exactly the same cover as* T. See the rightmost panel of Fig. 2 for an example of a vertical decomposition. We make the following observation:

Proposition 2. *Any tiling T of cardinality k has a vertical decomposition of cardinality at most $2k - 1$.*

The aim of computing the l-segmentation is to find the vertical decomposition of the underlying tiling. First, observe that if the underlying optimal k-tiling is vertically non-overlapping, the dynamic programming algorithm finds it directly (provided l is set appropriately). For other k-tilings, we know from Proposition 2 that their vertical decompositions correspond to segmentations with at most $2k+1$ segments (including possible empty segments before and after the leftmost and rightmost tile, respectively). For inputs with optimal noise-free tilings (i.e., tilings with precision equal to 1), we obtain the following:

Proposition 3. *Given an input matrix X containing a noise-free k-tiling T^* and the integer k, the dynamic programming algorithm finds a k'-tiling T', with* $\text{recall}(T') = \text{recall}(T^*)$*, and $k \leq k' \leq 2k - 1$.*

Since the `GlobalTile` algorithm is given the cardinality constraint k, it may not return the entire vertical decomposition of size $k' \geq k$. In the worst case all $2k - 1$ tiles in the vertical decomposition T' have exactly the same weight, and `GlobalTile` drops $k - 1$ of these, i.e., $\text{recall}(T) \geq \frac{k}{2k-1} \text{recall}(T^*)$. Hence, `GlobalTile` finds a tiling T with $\text{recall}(T) \geq \frac{1}{2} \text{recall}(T^*)$. Notice that in practice `GlobalTile` can perform even better, as it looks for the best k-tiling from all

segmentations *up to* size $2k+1$, and the vertical decomposition may in practice contain fewer than $2k-1$ tiles.

Propositions 2 and 3 above imply in practice that the `GlobalTile` algorithm can find tilings with recall very close to optimum, at the cost of using at most twice the amount of tiles used in an optimal solution.

4 Experiments

In this section we empirically investigate the relationship between (i) precision and recall and (ii) between recall and cardinality. We test the performance and behaviour of both tiling algorithms on four real and two synthetic datasets. We also compare the tilings we find for real data with tilings of a randomised version of the data having the same row marginals. The purpose of this comparison is to establish that the tilings we have found indeed reflect meaningful structure instead of noise.

The algorithms are implemented in R and are released as the `semigeom` R-package. This package and all source code used for the experiments is available for download[1].

4.1 Datasets

We consider four real and two synthetic datasets, described in detail below. Properties of the datasets are summarised in Table 1.

1. **Stock indices** The `stock indices` data contains daily stock exchange closing index data from 1995 to 2000[2] for 10 stock exchanges. We discretised the data into a two-letter alphabet by first calculating the gradient of the linear trend of each stock index in a 30-day windows and assigning 1 if the trend was rising (gradient positive) and -1 if the trend was decreasing (gradient negative). Since both labels are of interest, `stock indices (growth)` will denote the dataset when the label of interest is 1, and `stock indices (decline)` will denote the dataset when the label of interest is -1.
2. **Stock prices** The `stock prices` dataset contains daily closing prices between 03.01.2011–31.12.2015 (1257 trading days) for 400 randomly selected stocks. The data was discretised by calculating the change (in %) in closing price from the previous day. Instances for which the closing price dropped more than -6% were set to 1 and all other instances were set to zero.
3. **Paleo data** The `paleo` dataset is a binary 139×124 matrix describing the presence or absence of species (139) of Cenozoic mammals in different sites (124) [7,30]. The sites in the dataset have a temporal order determined by specialists.

[1] https://github.com/bwrc/semigeom-r.
[2] Data from http://www.economicswebinstitute.org/data/stockindexes.zip.

4. **EEG dataset** The `EEG` dataset is part of the ISRUC-Sleep dataset [18], consisting of EEG polysomnography recordings from 10 healthy people. There are six EEG channels located on the frontal (channels F3, F4), central (channels C3, C4), and occipital (channels O1, O2) parts of the scalp, all sampled at 200 Hz. We chose to use data from the first 6.5 hours of sleep, where data from all subjects was available, giving a total of 4680000 samples per channel. The data was discretised by calculating the EEG frequency spectrum in four-second windows, and for each window determining which of the following 5 frequency bands was dominant (delta: 1–4 Hz, theta: 4–7 Hz, alpha: 8–15 Hz, beta: 16–31 Hz, and gamma: 16–100 Hz). Each window was then assigned a letter corresponding to the dominant band. After the discretisation the data contains 5850 samples for each of the six channels.
5. **Synthetic data** The synthetic datasets are randomly generated binary matrices containing a given number of randomly inserted segments with a fraction p of ones. We used two types of synthetic datasets; (i) k randomly inserted tiles of size $l \times l$ and (ii) tiles formed from k randomly inserted varying-width segments spanning l rows.

Table 1. Properties of the datasets used in the experiments. Size is the number of rows and columns in the original data and the discrete size are the dimensions of the discretised data used as input to the algorithms. Density is the fraction of 1 s in the input matrix.

Dataset	Size	Size discrete	Density
Stock indices (growth)	10×1353	10×45	0.62
Stock indices (decline)	10×1353	10×45	0.37
Stock prices	400×1258	400×1257	0.02
Paleo	139×124	139×124	0.11
EEG subject 4	6×4680000	6×5850	0.14
EEG subject 7	6×4680000	6×5850	0.21
Synthetic 1	50×100	50×100	0.28
Synthetic 2	50×100	50×100	0.16

4.2 Basic Results

Precision-Recall Trade-Off. We first study the precision-recall trade-off of the algorithms. Precision-recall curves for two different cardinalities (3 and 7) are shown in Fig. 3 for the `synthetic1`, `paleo`, `stock prices`, and `stock indices (decline)`. The curves were computed by modifying the precision constraint parameter. The results suggest that both `MaxTile` and `GlobalTile` perform similarly in terms of balance between precision and recall.

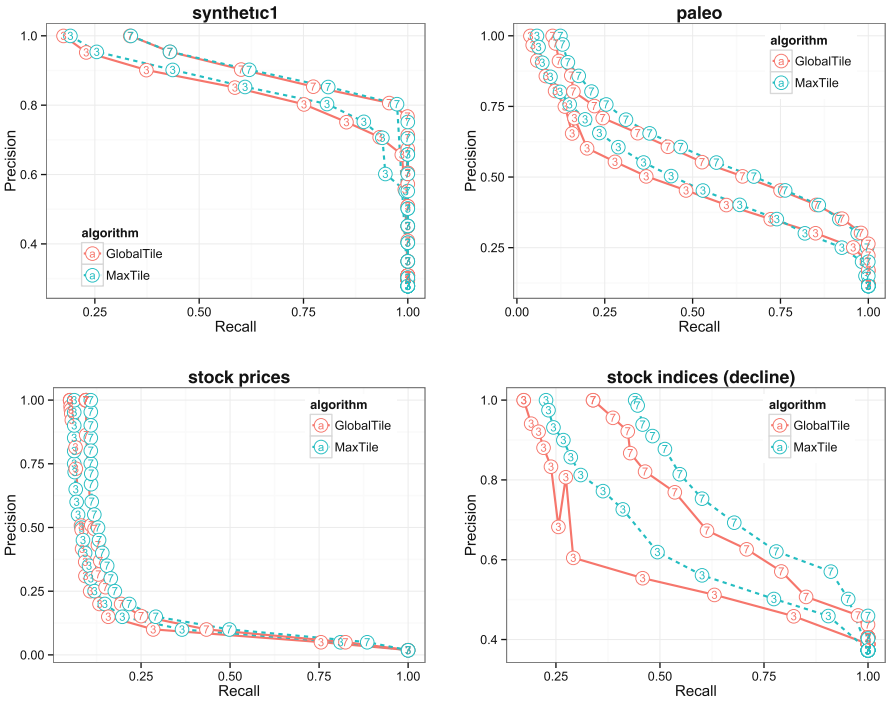


Fig. 3. Precision-recall curve for the algorithms for different datasets using cardinalities of 3 and 7.

Recall in Original vs. Randomised Data. To study if the found tilings reflect meaningful structure, we compare the recall of the tiling of the original data with the recall of a tiling for randomised data, using otherwise the same parameters. The randomised data are generated by shuffling the original values of every data row uniformly at random. This simple randomisation scheme breaks any temporal connections the event sequences may have, and thus any tiling found after randomisation only contains meaningless structure. Tilings found in the original unshuffled data should have a higher recall.

The cardinality-recall curves for the algorithms on the `paleo`, `stock prices`, `stock indices (growth)`, and `stock indices (decline)` datasets are shown in Fig. 4. The precision constraint was set to the value $P_{\min}$ in Table 2. The CR-curves show that for every dataset, both algorithms find a tiling with higher recall than what can be found in the randomised data. This indicates that tilings of the original data indeed contain meaningful structure. Moreover, although the algorithms perform quite similarly, `MaxTile` in general achieves a higher recall for a given cardinality, as expected, since the algorithm allows overlapping tiles. The performance of the algorithms on the `paleo` dataset is very similar. The structure of the dataset fits well with the assumption of vertical segments of

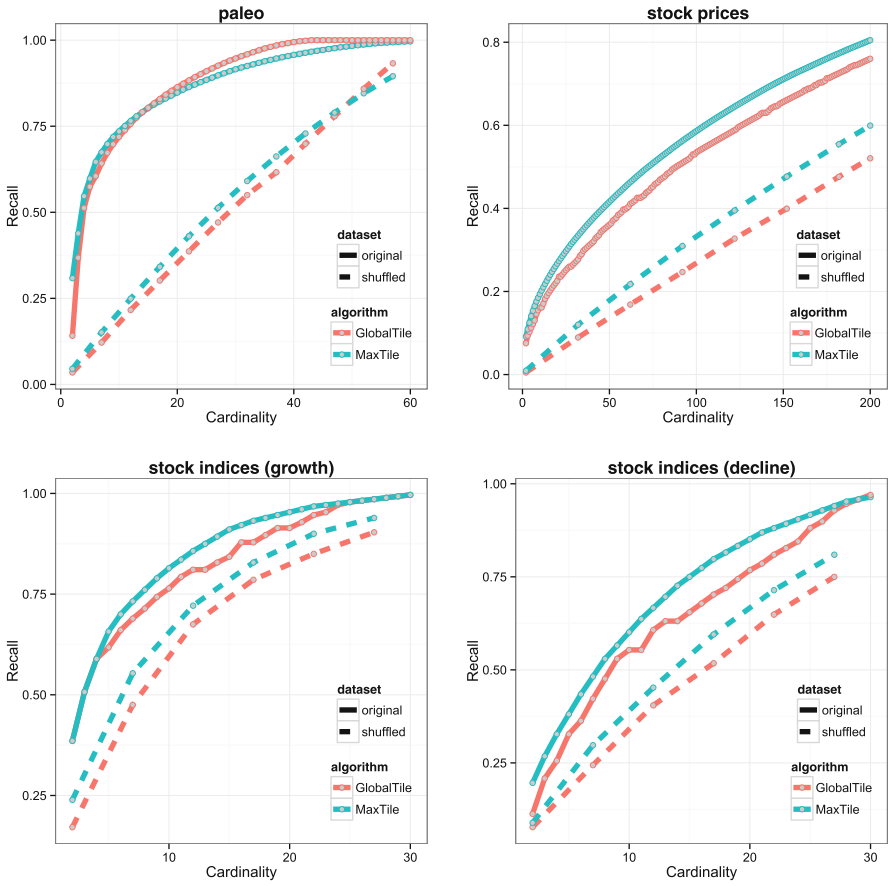


Fig. 4. Cardinality-recall curve for the algorithms for different datasets.

`GlobalTile` and hence `GlobalTile` achieves a higher recall than `MaxTile` for the same cardinality.

Running Times. Typical wall-clock running times (using unoptimised R-code running on a 1.8 GHz Intel Core i7 CPU) as well as examples of recall and cardinality for a given precision constraint for all datasets and both algorithms are shown in Table 2.

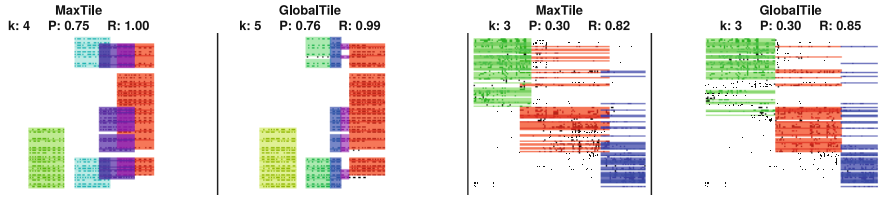
4.3 Example Tilings

Paleo dataset. The analysis results of the `synthetic1` and `paleo` datasets are shown in Fig. 5. Both algorithms manage to successfully capture the structure in the `synthetic1` dataset. Notice the difference in terms of selected tiles, due to allowance of overlapping tiles by `MaxTile`. The `paleo` dataset is very sparse,

Table 2. Numerical results from the experiments. P, R, and k stand for precision, recall, and cardinality, respectively. The time-column gives the calculation time in seconds.

Dataset	Algorithm	P_{min}	k_{max}	P	R	k	time [s]
`stock indices`	GlobalTile	0.85	5	0.87	0.62	5	0.12
(growth)	MaxTile	0.85	5	0.87	0.66	5	0.03
`stock indices`	GlobalTile	0.85	5	0.90	0.33	5	0.11
(decline)	MaxTile	0.85	5	0.91	0.38	5	0.03
`stock prices`	GlobalTile	0.30	10	0.32	0.16	10	112.90
	MaxTile	0.30	10	0.30	0.19	10	104.41
`paleo`	GlobalTile	0.30	3	0.30	0.85	3	1.32
	MaxTile	0.30	3	0.30	0.82	3	0.60
`EEG`	GlobalTile	0.60	10	0.60	0.79	10	1451.61
(alpha, subject 4)	MaxTile	0.60	10	0.60	0.79	10	461.38
`EEG`	GlobalTile	0.60	10	0.60	0.94	10	1456.06
(alpha, subject 7)	MaxTile	0.60	10	0.60	0.94	10	473.34
`synthetic1`	GlobalTile	0.75	5	0.76	0.99	5	0.47
	MaxTile	0.75	5	0.75	1.00	4	0.21
`synthetic2`	GlobalTile	0.75	5	0.80	1.00	5	0.48
	MaxTile	0.75	5	0.76	1.00	5	0.22

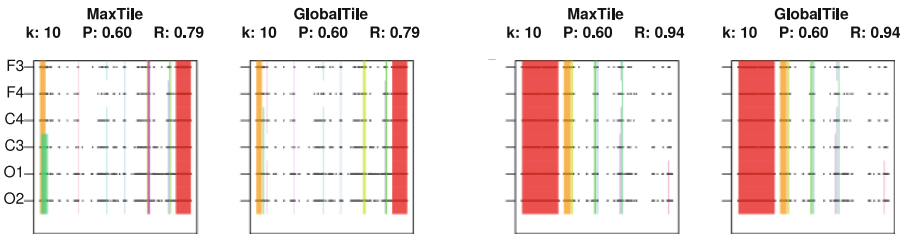
so a precision threshold of 0.3 was used. The structure in the three geological epochs in the data are clearly visible in the tilings and both algorithms achieve a high recall already with three tiles. The performance of the two algorithms is similar in terms of precision, and the discovered structure is also equivalent. The middle tile (in red) is wider for `MaxTile`, since the algorithm allows overlapping tiles.



(a) `synthetic1` (b) `paleo`

Fig. 5. Tiling of the `synthetic1` and `paleo` datasets. (Color figure online)

EEG dataset. We chose to consider only one letter corresponding to the alpha band from each channel and calculated the tilings with a maximal cardinality



(a) Subject 4 **(b) Subject 7**

Fig. 6. Tiling of the EEG activity in the alpha band for subjects 4 and 7. (Color figure online)

of 10 and a precision threshold of 0.60. This analysis, hence, reveals whether there are segments with similarly dense alpha activity on the different channels. It is expected that neighbouring channels in the same front-back location (frontal, central, and occipital) should be similar and channels in different front-back locations should be somewhat different. We omitted singleton tiles, i.e., tiles with a cardinality less than two, since we are interested in the relationships between different channels. Figure 6 shows the tiling obtained for the two subjects with prominent patterns. The `MaxTile` and `GlobalTile` tilings are very similar, with some small differences. In `MaxTile` tiling of subject 4 there is a relatively wide segment (green tile), where the neighbouring occipital channels and the C3 channel share a similar pattern of alpha activity. Both algorithms find a similar pattern of alpha activity in all channels of subject 4 (red tile at right end) and subject 7 (red tile at left end). Information regarding the relationships and spatiotemporal distributions of different EEG rhythms could be useful, e.g., when investigating the neural correlates of sleep patterns.

5 Discussion

In this paper we studied the problem of finding segments with similar properties in a collection of event sequences. We formalised the problem as a tiling problem and presented two algorithms for finding a tiling maximising the recall of a dataset discretised into an alphabet.

We empirically demonstrated the performance of the algorithms on real datasets from different domains; economics, palaeontology, and medicine, as well as on synthetic datasets. Using the methods described in this paper it is possible to uncover temporal phenomena in the data and the results provide meaningful insight into the interpretation of the underlying processes. The two algorithms emphasise slightly different properties of the data. For instance, the tilings found by the `MaxTile` algorithm consist of large, overlapping, submatrices satisfying the precision requirement, whereas the tilings produced by the `GlobalTile` algorithm are always non-overlapping.

Different datasets require different choices of parameters. The cardinality and precision threshold must be set depending on factors, such as the data density. As shown in the experimental evaluation, the proportion of the dataset explained by the tiles (i.e., the recall) varies depending on both the cardinality and precision. A reasonable heuristic for setting the precision constraint is to consider the overall data density, and set α to a smaller value so that the tiles indeed reflect denser regions in the input matrix.

Interestingly, we here showed that the special case of semigeometric tiling of a dataset, where we find a minimum full cover with perfect precision is tractable in polynomial time despite the related problems of combinatorial and fully geometric tiling being NP-hard. However, our main problem of covering a data matrix with a precision of $1 - \alpha$ with k tiles appears to be challenging, and we suspect it is **NP**-hard. The proof of this, however, remains an open question. Another important direction for future work is to consider the significance of the tilings found. In this paper we use a simple sanity check where the recall of the resulting tiling is compared against the recall of a tiling of the randomised data matrix. Methods to evaluate the significance of individual tiles are also of interest.

Acknowledgements. AH, AU, and KP were supported by Tekes (Revolution of Knowledge Work project) and Academy of Finland (decision 288814) and IK and PP by Swedish Foundation for Strategic Research (grant IIS11-0053).

References

1. Agrawal, R., Faloutsos, C., Swami, A.: Efficient similarity search in sequence databases. In: Lomet, D.B. (ed.) FODO 1993. LNCS, vol. 730, pp. 69–84. Springer, Heidelberg (1993). doi:10.1007/3-540-57301-1_5
2. Batal, I., Fradkin, D., Harrison, J., Moerchen, F., Hauskrecht, M.: Mining recent temporal patterns for event detection in multivariate time series data. KDD **2012**, 280–288 (2012)
3. Bellman, R.: On the approximation of curves by line segments using dynamic programming. Commun. ACM **4**(6), 284 (1961)
4. Chan, K.P., Fu, A.W.C.: Efficient time series matching by wavelets. ICDE **1999**, 126–133 (1999)
5. Cheng, Y., Church, G.M.: Biclustering of expression data. ISMB **8**, 93–103 (2000)
6. Faloutsos, C., Jagadish, H., Mendelzon, A.O., Milo, T.: A signature technique for similarity-based queries. SEQUENCES **1997**, 2–20 (1997)
7. Fortelius, M.: Coordinator: New and old worlds database of fossil mammals (NOW) (2016). University of Helsinki. http://www.helsinki.fi/science/now/
8. Franzblau, D.S., Kleitman, D.J.: An algorithm for covering polygons with rectangles. Inf. Control **63**(3), 164–189 (1984)
9. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-completeness. W.H. Freeman & Co, New York (1979)
10. Geerts, F., Goethals, B., Mielikäinen, T.: Tiling databases. In: Suzuki, E., Arikawa, S. (eds.) DS 2004. LNCS (LNAI), vol. 3245, pp. 278–289. Springer, Heidelberg (2004). doi:10.1007/978-3-540-30214-8_22
11. Gionis, A., Mannila, H., Seppänen, J.K.: Geometric and combinatorial tiles in 0–1 data. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) PKDD 2004. LNCS, vol. 3202, pp. 173–184. Springer, Heidelberg (2004). doi:10.1007/978-3-540-30116-5_18

12. Gionis, A., Mannila, H., Terzi, E.: Clustered segmentations. In: 3rd Workshop on Mining Temporal and Sequential Data, KDD 2004 (2004)
13. Györi, E.: A minimax theorem on intervals. J. Comb. Theor. Ser. B **37**(1), 1–9 (1984)
14. Hartigan, J.A.: Direct clustering of a data matrix. J. Am. Stat. Assoc. **67**(337), 123–129 (1972)
15. Hemminki, S., Nurmi, P., Tarkoma, S.: Accelerometer-based transportation mode detection on smartphones. In: SenSys 2013, p. 13 (2013)
16. Huang, C.F.: A hybrid stock selection model using genetic algorithms and support vector regression. Appl. Soft Comput. **12**(2), 807–818 (2012)
17. Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S.: Dimensionality reduction for fast similarity search in large time series databases. Knowl. Inf. Syst. **3**(3), 263–286 (2001)
18. Khalighi, S., Sousa, T., Santos, J.M., Nunes, U.: Isruc-sleep: a comprehensive public dataset for sleep researchers. Comput. Methods Program. Biomed. **124**, 180–192 (2015)
19. Klemettinen, M., Mannila, H., Toivonen, H.: Rule discovery in telecommunication alarm data. J. Network Syst. Manage. **7**(4), 395–423 (1999)
20. Knuth, D.E.: Irredundant intervals. J. Exp. Algorithmics (JEA) **1** (1996)
21. Kontonasios, K.N., De Bie, T.: An information-theoretic approach to finding informative noisy tiles in binary databases. In: SDM 2010, p. 153 (2010)
22. Lam, H.T., Pei, W., Prado, A., Jeudy, B., Fromont, É.: Mining top-k largest tiles in a data stream. In: ECML PKDD 2014, pp. 82–97. Springer, Heidelberg (2014)
23. Lee, J., Lee, Y., Jun, C.H.: A biclustering method for time series analysis. Ind. Eng. Manage. Syst. **9**, 129–138 (2010)
24. Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J.M., Glance, N.S.: Cost-effective outbreak detection in networks. KDD **2007**, 420–429 (2007)
25. Lin, J., Keogh, E., Lonardi, S., Chiu, B.: A symbolic representation of time series, with implications for streaming algorithms. SIGMOD **2003**, 2–11 (2003)
26. Lionello, P.: The climate of the venetian and north adriatic region: variability, trends and future change. Phys. Chem. Earth Parts A/B/C **40**, 1–8 (2012)
27. Madeira, S.C., Oliveira, A.L.: A linear time biclustering algorithm for time series gene expression data. In: Casadio, R., Myers, G. (eds.) WABI 2005. LNCS, vol. 3692, pp. 39–52. Springer, Heidelberg (2005). doi:10.1007/11557067_4
28. Nemhauser, G.L., Wolsey, L.A., Fisher, M.L.: An analysis of approximations for maximizing submodular set functionsi. Math. Program. **14**(1), 265–294 (1978)
29. Patel, P., Keogh, E., Lin, J., Lonardi, S.: Mining motifs in massive time series databases. ICDM **2002**, 370–377 (2002)
30. Puolamäki, K., Fortelius, M., Mannila, H.: Seriation in paleontological data using markov chain monte carlo methods. PLoS Comput. Biol. **2**(2), e6 (2006)
31. Rakthanmanon, T., Keogh, E.J., Lonardi, S., Evans, S.: MDL-based time series clustering. Knowl. Inf. Syst. **33**(2), 371–399 (2012)
32. Ukkonen, A.: Mining local correlation patterns in sets of sequences. In: Gama, J., Costa, V.S., Jorge, A.M., Brazdil, P.B. (eds.) DS 2009. LNCS (LNAI), vol. 5808, pp. 347–361. Springer, Heidelberg (2009). doi:10.1007/978-3-642-04747-3_27
33. Xiang, Y., Jin, R., Fuhry, D., Dragan, F.F.: Succinct summarization of transactional databases: An overlapped hyperrectangle scheme. Knowl. Discov. Data Min. **23**, 758–766 (2008)
34. Zhang, Y., Zha, H., Chu, C.H.: A time-series biclustering algorithm forrevealing co-regulated genes. Int. Conf. Informationtechnology: Coding Comput. **1**, 32–37 (2005)