Continuous Imputation of Missing Values in Streams of Pattern-Determining Time Series

Kevin Wellenzohn   Michael H. Böhlen
Department of Computer Science
University of Zurich, Switzerland
{wellenzohn, boehlen}@ifi.uzh.ch

Anton Dignös   Johann Gamper
Hannes Mitterer
Faculty of Computer Science
Free University of Bolzano, Italy
firstname.lastname@unibz.it

ABSTRACT

Time series data is ubiquitous but often incomplete, e.g., due to sensor failures and transmission errors. Since many applications require complete data, missing values must be imputed before further data processing is possible.

We propose Top-k Case Matching (TKCM) to impute missing values in streams of time series data. TKCM defines for each time series a set of reference time series and exploits similar historical situations in the reference time series for the imputation. A situation is characterized by the *anchor point* of a pattern that consists of l consecutive measurements over the reference time series. A missing value in a time series s is derived from the values of s at the anchor points of the k most similar patterns. We show that TKCM imputes missing values *consistently* if the reference time series *pattern-determine* time series s, i.e., the pattern of length l at time t_n is repeated at least k times in the reference time series and the corresponding values of s at the anchor time points are similar to each other. In contrast to previous work, we support time series that are not linearly correlated but, e.g., phase shifted. TKCM is resilient to consecutively missing values, and the accuracy of the imputed values does not decrease if blocks of values are missing. The results of an exhaustive experimental evaluation using real-world and synthetic data shows that we outperform the state-of-the-art solutions.

1. INTRODUCTION

Time series data appears in many application domains, e.g., meteorology, sensor networks, the financial world, and network monitoring. Often time series data is incomplete with values missing because of sensor failures, transmission errors, etc. Many applications require complete data, hence missing values must be recovered before further data processing is possible.

Our research is motivated by the problem of missing values in the data collected by the *Südtiroler Beratungsring für Obst- und Weinbau* (SBR). The SBR monitors and analyzes meteorological data streams in real time and alerts wine and apple farmers of potential harvest threats, such as frost, apple scab, and fire blight. The SBR operates a network of more than 130 weather stations in South Tyrol, each of which records approximately 20 meteorological parameters at a sample rate of five minutes. The measurements date back to 2007 with a total of 88.9M measurements. For illustration purposes, we use the temperature taken at one meter above ground level, which ranges from -20.3°C to $+40.3$°C and has a total of 7.8M ($= 8\%$) missing values. Currently, missing values are manually imputed by domain experts, based on the values at neighboring stations.

Various works have observed that time series are correlated and imputation techniques have been proposed that exploit the information of co-evolving time series [12, 13, 14, 16, 25]. Popular solutions include SVD based matrix decomposition techniques [11, 12], multivariate autoregression analysis [25], and PCA (principal component analysis) guided data summarization [16, 17, 23]. These approaches perform well if the time series are linearly correlated according to the Pearson correlation. The imputation accuracy deteriorates if the time series are shifted and have a Pearson correlation close to zero.

In this paper, we propose Top-k Case Matching (TKCM) to impute missing values in streams of non-linearly correlated time series. TKCM defines for each time series s a small set $\mathbf{R}_s$ of *reference time series*. If the value in s at the current time t_n is missing, TKCM defines a *query pattern* $P(t_n)$ that is *anchored* at t_n and composed of the l most recent measurements of the reference time series. Then, the k most similar non-overlapping patterns to the query pattern within a given time window are determined. The missing value is derived from the values of time series s at the anchor points of the k most similar patterns. This process is illustrated in Fig. 1, where the value of the time series s at the current time t_n is missing (small circle on the right). There are two reference time series of s, i.e., $\mathbf{R}_s = \{r_1, r_2\}$. The query pattern $P(t_n)$ is composed of the snippets of the reference time series in the black frame. The $k = 2$ most similar patterns to the query pattern are anchored at t_i and t_j and are shown as dashed rectangles. The missing value of s is derived from the values of s at the anchor points t_i and t_j (small circles).

TKCM exploits two common properties of time series. First, time series often exhibit (not necessarily regularly) repeating patterns, also referred to as seasonal patterns. Second, time series are (not necessarily linearly) correlated in the sense that, whenever a pattern in a set of reference time series repeats, time series s exhibits similar values. If these two properties are satisfied we say that at time t_n the reference time series $\mathbf{R}_s$ *pattern-determine* time series s, denoted by $\mathbf{R}_s, t_n \xrightarrow{\text{pd}} s$. In other words, whenever similar patterns occur in $\mathbf{R}_s$, the values of time series s are similar to each other, too. In contrast to previous work, this property allows not only linearly correlated reference time series but permits phase shifts. For instance, in Fig. 1 the two (shifted) reference time se-

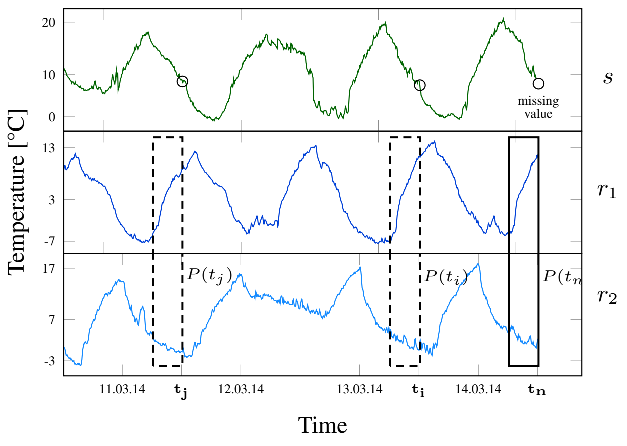


Figure 1: Imputation of a missing value of s at time t_n.

ries $\mathbf{R}_s = \{r_1, r_2\}$ *pattern-determine* time series s at time t_n. We show that TKCM imputes missing values consistently if the time series are pattern determining.

The paper makes the following technical contributions:

- We present and formalize Top-k Case Matching (TKCM) to impute missing values in streams of pattern-determining time series, which covers non-linear relationships between time series.
- We show that TKCM computes correct results for shifted time series that are not linearly correlated. We use a pattern of length $l > 1$ to exploit several consecutive measurements to find similar historical situations.
- We propose a dynamic programming scheme to find the k non-overlapping patterns that minimize the sum of dissimilarities with respect to the query pattern.
- We empirically show on real-world and synthetic datasets that TKCM: (a) outperforms state-of-the-art solutions, (b) can impute values in time series with phase shifts, and (c) is resilient to large blocks of consecutively missing values.

The paper is structured as follows. Section 2 discusses related works. After the preliminaries in Section 3 we describe our approach in Section 4. Section 5 analyzes the properties of TKCM and works out the differences between linear and non-linear correlations. In Section 6 we provide an implementation of TKCM that uses a dynamic programming solution to find the k non-overlapping patterns that are most similar to the query pattern. We continue with the experimental evaluation in Section 7 before we conclude this paper and present future research directions in Section 8.

2. RELATED WORK

The need to recover missing data arises in many applications, ranging from meteorology [18, 26], to social science [20], machine learning [2], motion capture systems [14] and DNA microarray analysis [24]. Imputing a missing value means to recover it with a good estimate that is derived from intrinsic relationships in the underlying dataset.

Simple imputation techniques include *mean* and *mode* imputation [2], which replace the missing value with the mean or mode of the same attribute. Interpolation techniques, such as linear interpolation and spline interpolation, estimate the missing value from immediately preceding and succeeding values of the same attribute. If the gap is long, i.e., if many consecutive values are missing, these interpolation techniques perform badly. For instance, if an entire period of a sine wave is missing, linear interpolation would replace the gap with a straight line. Regression methods [25] estimate the missing value of a time series (e.g., temperature in Zurich) based on the value of other time series (e.g., temperature in Bern and Basel). Paulhus et al. [18] observed that nearby weather stations have similar values and computed a missing value at one station as the average of the values of nearby stations. Yozgatligil et al. [26] give a recent survey of imputation methods for meteorological time series and cover approaches based on neural networks and multiple imputation [19].

The ARIMA model [4] is a popular time series forecasting model that is a generalization of the auto-regressive (AR) model. ARIMA assumes a linear dependency of unknown future values on known past values of the time series. Finding the proper values for p, d, q in an ARIMA(p, d, q) model is tedious, complex and involves manual analysis, known as the Box-Jenkins methodology [4].

Batista et al. [2] study the problem of missing data in the context of machine learning algorithms and present the k-Nearest Neighbor Imputation (kNNI) method to recover these values. For a multi-attribute object (e.g., breast cancer test with multiple measurements) that has a missing value for one attribute A, the kNNI approach looks for k objects with similar values for the other attributes according to a distance metric that is not specified. The missing value is derived from the values of attribute A in these objects. Troyanskaya et al. [24] extend kNNI to weight the k most similar items according to their similarity. Our approach, TKCM, uses the concept of nearest neighbors (k most similar patterns), but is designed for time series streams and uses a two-dimensional query pattern for which the k most similar *non-overlapping* patterns according to the Euclidean distance are searched.

Khayati et al. [11] propose REBOM to recover blocks of missing values in irregular time series with non-repeating trends. The algorithm builds a matrix which stores the incomplete time series and the n most linearly correlated time series according to Pearson correlation. Missing values are first initialized, e.g., using linear interpolation. Then the matrix is iteratively decomposed using the *Singular Value Decomposition* (SVD) method, where the least significant singular values are truncated. Due to the quadratic runtime complexity, REBOM does not scale to long time series. Next, Khayati et al. [12] present a solution with linear space complexity based on the *Centroid Decomposition* (CD), which is an approximation of SVD. Unlike our approach, SVD and CD assume a linear correlation between an incomplete time series and its reference time series. If time series are not linearly correlated, the imputation accuracy deteriorates since these trends are captured by the truncated least significant singular values. Khayati et al. [13] show that CD imputes more accurately than SVD when some reference time series are shifted and hence lowly linearly-correlated, because CD prioritizes highly linearly-correlated reference time series. Nevertheless, their experiments show that adding more lowly-correlated reference time series has a negative impact on CD's accuracy.

Sorjamaa et al. [15, 22] propose an imputation method based on a Self-Organizing-Map (SOM), which is an unsupervised learning technique based on neural networks. A combination of SVD and SOM [22] uses SVD for the imputation after initializing missing values in the matrix by a SOM classifier, whereas [15] combines two SOM classifiers for the imputation. Both methods are only evaluated on linearly correlated time series.

DynaMMo [14] is used for mining, summarizing, and imputing time series extracted from human motion capture systems. It is based on Kalman filters, which, similar to SVD, assume a linear correlation between time series to accurately estimate unknown values. Moreover, unlike our approach, DynaMMo allows only one reference time series, which often is insufficient for an accurate imputation.

The works most similar to our approach are MUSCLES [25] and SPIRIT [16, 17, 23], which focus on the online imputation of missing values in streams of time series data. Both algorithms use variants of auto-regressive (AR) models and exploit linear correlations between data streams. When the linear correlation diminishes, as in the case of shifted time series, none of the two approaches performs well.

MUSCLES [25] is an online algorithm that is based on a *multivariate* auto-regression model, whose parameters are incrementally updated using the Recursive Least Squares method. Besides past values of the incomplete time series, MUSCLES takes also the most recent values of co-evolving and linearly correlated time series into account that are within a window p. How to choose p is not discussed; in the experiments $p = 6$ is used. After p consecutive missing values, MUSCLES relies exclusively on imputed values for the incomplete time series. Since small imputation inaccuracies accumulate over a long stretch of missing values, MUSCLES accuracy deteriorates. Additionally, MUSCLES does not scale well to a large number of streams, unless an expensive offline subset selection on the time series is performed [17].

SPIRIT [16, 17, 23] uses an online Principal Component Analysis (PCA) to reduce a set of n co-evolving and correlated streams to a small number of k hidden variables that summarize the most important trends in the original data. For each hidden variable, SPIRIT fits one AR model on past values, which is incrementally updated as new data arrives. If a value is missing, the AR models are used to forecast the current value of each variable, from which an estimate of the missing value is derived. The imputed value, along with the non-missing values, is then used to update the forecasting models. Updating the models with imputed models incurs similar problems as MUSCLES since inaccuracies are propagated. Since PCA and SVD are based on the same underlying principle, PCA shares SVD's weaknesses for shifted time series.

From an implementation perspective, TKCM needs to find similar patterns in time series. This problem has been studied extensively for a single time series, yielding different dimensionality reduction techniques, e.g., Discrete Fourier Transform [6], Piecewise Aggregate Approximation [7], and iSAX [21]. Keogh et al. [10] present a fast approach to find a subsequence (i.e., one-dimensional pattern), termed shapelet, of a time series that is most representative for a set of time series. Finding patterns in our approach is more complex. We seek patterns that span several time series, and we have to select the k most similar non-overlapping patterns. The main focus of this work is not performance, but an accurate imputation of shifted time series streams.

3. PRELIMINARIES

Consider a set $\mathbf{S} = \{s_1, s_2, \ldots\}$ of streaming time series. Each time series reports values from a sensor measured at time points $\ldots, t_{n-2}, t_{n-1}, t_n$, where t_n denotes the current time, i.e., the time of the latest measurement. The value of a time series $s \in \mathbf{S}$ at time t_i is denoted as $s(t_i)$. We write $s(t_n) = \text{NIL}$ to denote that the current value of s is missing. $W = \{t_{n-L+1}, \ldots, t_{n-1}, t_n\}$ denotes the L time points in our streaming window for which we keep measurements in main memory. We assume that the streaming window W is long enough to include the query pattern and k non-overlapping similar patterns.

For each time series $s \in \mathbf{S}$ there exists an ordered sequence $\langle r_1, r_2, \ldots\rangle$ of candidate reference time series, where $r_i \in \mathbf{S} \setminus \{s\}$. They have been identified by domain experts and are consulted if the current value in s is missing and must be recovered. The candidate reference time series of s are ranked according to how suitable they are for imputing a missing value in s. A single reference time series does not yield a robust method to estimate a missing value. Instead the d best candidate reference time series that do not have a missing value at the current time t_n are used. Let $s \in \mathbf{S}$ be an incomplete time series with $s(t_n) = \text{NIL}$. The *reference time series* $\mathbf{R}_s$ for s at the current time t_n are the first d time series in the ordered sequence for which $r(t_n) \neq \text{NIL}$.

Note that there can be multiple incomplete time series with a missing value at t_n. For each incomplete time series s_i its missing value $s_i(t_n)$ is imputed individually using the respective set of reference time series $\mathbf{R}_{s_i}$.

Example 1. As a running example, we use the four time series in Table 2. The current time is t_n = 14:20. Time series s is incomplete, hence the missing value at 14:20 must be imputed. We assume a sliding window of one hour, containing $L = 12$ measurements. For all time points before t_n the values either have been reported by the sensor or have been imputed, e.g., $r_2(\text{13:40}) = \widehat{18.8}$°C. The candidate reference time series are $\langle r_1, r_2, r_3\rangle$. At the current time t_n = 14:20, the $d = 2$ reference time series for s are $\mathbf{R}_s = \{r_1, r_2\}$. When the current time was t_n = 13:40, we had $\mathbf{R}_s = \{r_1, r_3\}$ since $r_2(\text{13:40})$ was missing. □

Notation	Description
t_n	Current time
$\mathbf{S} = \{s_1, s_2, \ldots\}$	Set of time series
$s(t_n) = \text{NIL}$	Missing value of time series s at time t_n
$\hat{s}(t_n) \neq \text{NIL}$	Imputed value of time series s at time t_n
d	Number of reference time series
$\mathbf{R}_s = \{r_1, \ldots, r_d\}$	Set of d reference time series for s
$W = \{\ldots, t_n\}$	Time points in streaming window
L	Length of streaming window W
l	Pattern length
$P(t_i)$	Pattern anchored at time t_i
k	Number of anchor points
$\mathbf{A} = \{t_{i_1}, \ldots, t_{i_k}\}$	k most similar anchor points

Table 1: Summary of notation.

4. TOP-K CASE MATCHING (TKCM)

4.1 Approach

For the recovery of a missing value in an incomplete time series we look for patterns in the past when the values of the reference time series were similar to the current values.

Definition 1. (*Pattern*) Let $\mathbf{R}_s = \{r_1, \ldots, r_d\}$ be the reference time series for an incomplete time series s. The *pattern* $P(t_i)$ of length $l > 0$ over $\mathbf{R}_s$ that is anchored at time t_i is defined as a $d \times l$ matrix $P(t_i)$ as follows:

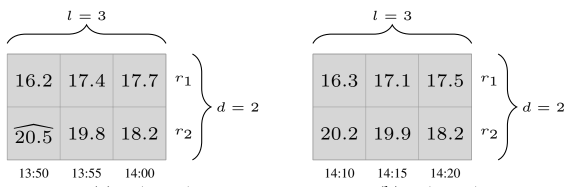
$$\begin{aligned} P(t_i) = ((&r_1(t_{i-l+1}), \ldots, r_1(t_i)),\\ &\vdots \qquad\qquad\quad \vdots\\ (&r_d(t_{i-l+1}), \ldots, r_d(t_i))). \end{aligned}$$

Time t	$\cdots$	13:25	13:30	13:35	13:40	13:45	13:50	13:55	14:00	14:05	14:10	14:15	14:20
s	$\cdots$	22.8°C	21.4°C	21.8°C	$\widehat{23.1}$°C	23.5°C	22.8°C	21.2°C	21.9°C	23.5°C	22.8°C	21.2°C	NIL
r_1	$\cdots$	16.5°C	17.2°C	17.8°C	16.6°C	15.8°C	16.2°C	17.4°C	17.7°C	15.3°C	16.3°C	17.1°C	17.5°C
r_2	$\cdots$	20.3°C	19.8°C	18.6°C	$\widehat{18.8}$°C	$\widehat{20.0}$°C	$\widehat{20.5}$°C	19.8°C	18.2°C	20.1°C	20.2°C	19.9°C	18.2°C
r_3	$\cdots$	14.0°C	14.8°C	13.6°C	13.0°C	14.5°C	14.3°C	14.0°C	15.0°C	13.0°C	14.5°C	14.3°C	14.6°C

Table 2: Time series s with a missing value at time t_n = 14:20 and the three reference time series r_1, r_2 and r_3.

A pattern is anchored at a time point t_i and consists of the values from t_{i-l+1} to t_i of each reference time series. Each row represents a subsequence of a reference time series, and each column represents the values of the reference time series at a time point. The pattern contains for each reference time series only the values at time t_i, if $l = 1$. The pattern includes additionally the preceding $l - 1$ values, and hence captures the trend, if $l > 1$.

Example 2. Figure 2 shows two patterns over the reference time series $\mathbf{R}_s = \{r_1, r_2\}$ in our running example (cf. Table 2). Both patterns have length $l = 3$ and are anchored at time points 14:00 and 14:20, respectively. Pattern P(14:00) contains one imputed value, namely r_2(13:50) = $\widehat{20.5}$°C. Since $l > 1$ the pattern captures the current trend of the time series. □



(a) P(14:00) (b) P(14:20)

Figure 2: Two patterns of length $l = 3$ over $d = 2$ reference time series.

The pattern that is anchored at the current time is termed *query pattern* $P(t_n)$. We search in the reference time series for the k patterns that are most similar to $P(t_n)$ using the L_2 norm.

Definition 2. (*Pattern Dissimilarity*) Let s be an incomplete time series with reference time series $\mathbf{R}_s$ at time t_n. The *dissimilarity*, δ, between two patterns $P(t_m)$ and $P(t_n)$ is defined as

$$\delta(P(t_m), P(t_n)) = \sqrt{\sum_{r_i \in \mathbf{R}_s} \sum_{0 \le j < l} (r_i(t_{m-j}) - r_i(t_{n-j}))^2}.$$

Example 3. The dissimilarity between the two patterns in Figure 2 is computed as follows: $\delta(P(\text{14:00}), P(\text{14:20}))$ = $\sqrt{(17.7 - 17.5)^2 + (17.4 - 17.1)^2 + (16.2 - 16.3)^2 + \ldots}$ = 0.43. □

The dissimilarity measure is used to determine the k most similar patterns to query pattern $P(t_n)$. The anchor time points of these k patterns are referred to as the k most similar anchor points $\mathbf{A}$.

Definition 3. (*k Most Similar Anchor Points*) Let $P(t_n)$ be the query pattern for incomplete time series s at time t_n with reference time series $\mathbf{R}_s$, and L be the length of the streaming time window. The k most similar *anchor points* to t_n are a set $\mathbf{A} \subseteq W$, with $|\mathbf{A}| = k$, for which the following holds:

$$\forall t \in \mathbf{A} : t_{n-L+l} \le t \le t_{n-l} \tag{1}$$

$$\forall t, t' \in \mathbf{A} : t \ne t' \rightarrow |t - t'| \ge l \tag{2}$$

$$\forall \mathbf{A}' : (1) \wedge (2) \wedge |\mathbf{A}'| = k \rightarrow \sum_{t_i \in \mathbf{A}} \delta(P(t_i), P(t_n)) \le \sum_{t_i \in \mathbf{A}'} \delta(P(t_i), P(t_n)) \tag{3}$$

The first condition states that all patterns are within the time window and do not overlap $P(t_n)$. The second condition states that the patterns do not overlap each other. The third condition ensures that the patterns that are anchored at the time points in $\mathbf{A}$ minimize the sum of the dissimilarities with respect to query pattern $P(t_n)$.

We pick only *non-overlapping* patterns to avoid near duplicates [5, 8]. Our experiments have shown that if overlapping patterns were allowed, the k most similar anchor points for some pattern $P(t_i)$ are frequently time points t_{i+1} and t_{i-1}, which anchor the first and second most similar patterns, etc. This is clearly not desired. Instead, non-overlapping patterns guarantee that we find a diverse set of patterns on which the imputation is based.

The missing value in the incomplete time series s is the average of the values of s at the most similar time points.

Definition 4. (*Imputed Value*) Let s be a time series with reference time series $\mathbf{R}_s$ at t_n and missing value $s(t_n)$. Furthermore, let $\mathbf{A}$ be the k most similar time points to the current time. The *imputed value* $\hat{s}(t_n)$ for time series s at time t_n is

$$\hat{s}(t_n) = \frac{1}{k} \sum_{t \in \mathbf{A}} s(t). \tag{4}$$

Example 4. Figure 3 shows a graphical representation of our running example (cf. Table 2). The value of s at time t_n = 14:20 is missing and must be imputed. The query pattern P(14:20) is framed in black. The two patterns most similar to the query pattern are shown as dashed rectangles and are anchored at time 14:00 and 13:35, respectively. Thus, $\mathbf{A}$ = {14:00, 13:35} are the anchor points. The missing value is computed as the average of the values s(14:00) and s(13:35): $\hat{s}$(14:20) = (21.9°C + 21.8°C)/2 = 21.85°C. □

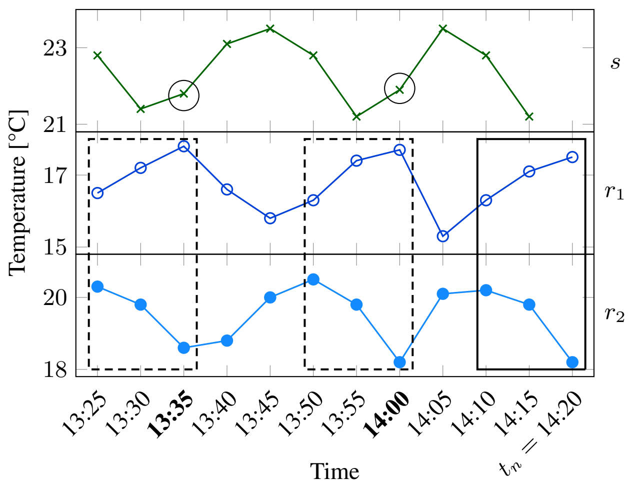


Figure 3: The k = 2 most similar non-overlapping patterns for query pattern P(14:20) are P(14:00) and P(13:35).

5. ANALYSIS

5.1 Correlation

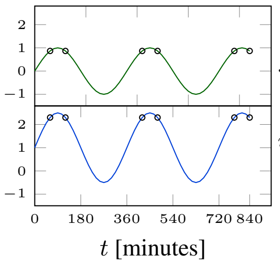
A salient property of TKCM is its ability to handle time series that are shifted and hence not linearly correlated. The Pearson correlation, the most common correlation measure, quantifies the degree of linear correlation between time series s and r, as

$$\rho(s,r) = \frac{\sum_{t\in W}(s(t)-\bar{s})(r(t)-\bar{r})}{\sqrt{\sum_{t\in W}(s(t)-\bar{s})^2}\sqrt{\sum_{t\in W}(r(t)-\bar{r})^2}},$$

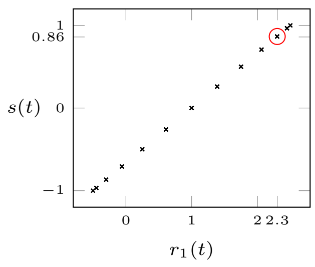
where $\bar{s}$ and $\bar{r}$ are the means of, respectively, s and r in window W. Pearson correlation ranges from -1 to 1, indicating total negative and positive correlation, respectively. Thus, s and r are linearly correlated if $|\rho(s,r)|$ is high. If $\rho(s,r) = 0$ time series s and r are not linearly correlated.

Intuitively, a linear correlation ensures that (a) if one time series has close values for two time points, also the other has close values for these two time points and (b) if one time series has far apart values for two time points, also the other has far apart values for these two time points.

Example 5. Consider Figure 4a with time series $s(t) = \text{sind}(t)$ and $r_1(t) = 1.5 \times \text{sind}(t) + 1$, having different amplitudes and offsets. The value of r_1 at $t = 840$ is $r_1(840) = 2.3$, and the same value $r_1(t)$ appears for time points $t \in \{780, 480, 420, 120, 60\}$. Figure 4a illustrates that these are exactly the time points for which s has the same value of $s(840) = 0.86$. Thus, the time series are perfectly linearly correlated. Figure 4b uses a scatterplot to display the correlation between s and r_1. The scatterplot displays for each time point t the point $(r_1(t), s(t))$. For instance, at time $t = 840$ we have $r_1(840) = 2.3$ and $s(840) = 0.86$ and the point $(2.3, 0.86)$ is displayed in the scatterplot. The more the scatterplot resembles a line with a non-zero slope, the higher the linear correlation and hence Pearson correlation. □



(a) Time series s and r_1



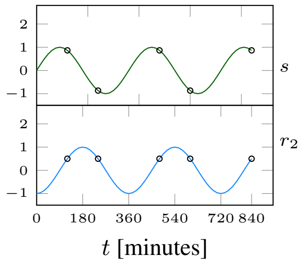
(b) Scatterplot of linear correlation

Figure 4: Linearly correlated time series $s(t) = \text{sind}(t)$ and $r_1(t) = 1.5 \times \text{sind}(t) + 1$.

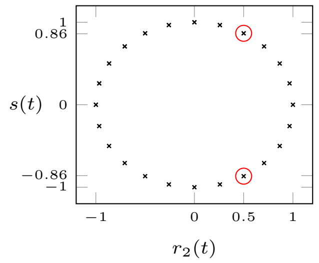
Example 5 illustrates how the imputation for linearly correlated time series works. If $s(t_n)$ is missing, we know that whenever time series r_1 observes value $r_1(t_n)$ (e.g., 2.3), time series s observes the same value $s(t)$ (e.g., 0.86). Hence we can use value $s(t)$ for any t where $r_1(t) = 2.3$ to impute value $s(t_n)$.

In contrast, if the Pearson correlation approaches zero, s can have very different values although the reference time series has the same value. This is illustrated in Example 6.

Example 6. Figure 5a depicts the time series $s(t) = \text{sind}(t)$ and $r_2(t) = \text{sind}(t-90)$. The two time series have the same amplitude and offset but they are phase shifted. Time series r_2 has the value $r_2(840) = 0.5$ also for time points $t \in \{600, 480, 240, 120\}$. However, s has different values, i.e., value $s(t) = 0.86$ for time points $t \in \{480, 120\}$ and value $s(t) = -0.86$ for time points $t = \{600, 240\}$. The scatterplot in Figure 5b shows that the data points do not cluster around a line, which means that they are non-linearly correlated. Their Pearson correlation is -0.0085. Note that for the same value of $r_2(t)$ we can have two different values for $s(t)$. For instance, for $r_2(t) = 0.5$ we have either $s(t) = 0.86$ or $s(t) = -0.86$. □



(a) Time series s and r_2



(b) Scatterplot of non-linear correlation

Figure 5: Non-linearly correlated time series $s(t) = \text{sind}(t)$ and $r_2(t) = \text{sind}(t-90)$

Example 6 illustrates the key problem with non-linear correlations: the values of one time series can no longer be used to reliably determine a missing value in another time series. In the experiments we will see that this leads to imputations with a high root mean square error.

5.2 Pattern Length

The previous section illustrated that shifted time series that are not linearly correlated are difficult to handle. Intuitively, for shifted time series it is not sufficient to consider a single point in time. Instead it is necessary to consider a pattern that includes neighboring time points to correctly relate time series. This section illustrates that a pattern with a length $l > 1$ improves the imputation for non-linear correlations. We write $P_l(t)$ to denote a pattern of length l anchored at time t.

Example 7. Figure 6 displays s and, for each time point t, the pattern dissimilarity of the pattern anchored at $r_1(t)$ to the query pattern $P(840)$, i.e., $\delta(P(t), P(840))$. Figure 6a does this for pattern length $l = 1$. The pattern dissimilarity is zero whenever the value of s is equal to $s(840)$. Figure 6b shows what happens if we increase the pattern length to $l = 60$. Also for this case whenever the pattern dissimilarity for the reference time series r_1 is zero, i.e., for 480 and 120, we have value 0.86 for time series s. Observe that for increasing values of l less patterns with distance zero exist (e.g., two in Fig. 6b instead of 5 in Fig. 6a). But the patterns with $l > 1$ at distance zero describe the situation better: $s(840)$ is located at a down-slope, and in Fig. 6b values of s where the pattern distance is zero only exist at down-slopes, while in Fig. 6a we have such values at both up- and down-slopes. □

Example 8. For shifted time series a pattern length $l > 1$ in addition captures the trend of time series and yields a more accurate imputation. First, Figure 7a illustrates s and the pattern dissimilarity to r_2 for $l = 1$. Figure 7b shows the same setting with $l = 60$. With $l > 1$ the pattern dissimilarity reaches zero only for time points 480 and 120, where time series s has value 0.86, which is the expected value for missing value $s(840)$. This illustrates that by increasing l, TKCM finds anchor points where the incomplete

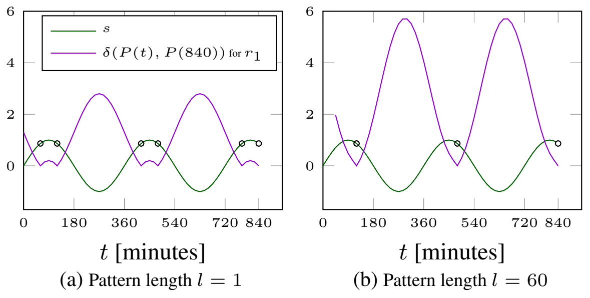


Figure 6: A longer pattern reduces the number of patterns that are identical to the query pattern.

time series s has similar values and trends. Consequently, TKCM uses pattern length l to effectively deal with shifted time series that are not linearly correlated. □

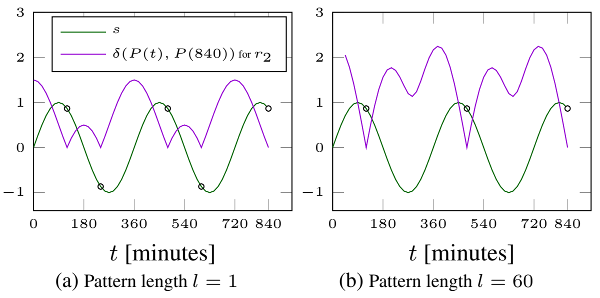


Figure 7: For shifted time series a longer pattern finds historical situations that are similar in value and trend.

LEMMA 5.1. *(*Monotonicity in Pattern Length*) The number of patterns that are within a distance τ to query pattern $P(t_n)$ decreases as the pattern length l increases:*

$$|\{t_m|\delta(P_{l+1}(t_m), P_{l+1}(t_n))\leq\tau\}| \leq |\{t_m|\delta(P_l(t_m), P_l(t_n))\leq\tau\}|$$

PROOF. Let $\Delta = \delta(P_l(t_m), P_l(t_n))$. If pattern length l is increased we get $\delta(P_{l+1}(t_m), P_{l+1}(t_n)) = \sqrt{\Delta^2 + \sum_{r_i\in\mathbf{R}_s}(r_i(t_{m-l}) - r_i(t_{n-l}))^2}$. Observe that both terms under the square root are non-negative, hence δ is monotonically increasing as l increases. It follows that the number of patterns with distance $\leq \tau$ decreases as l grows. □

5.3 Consistent Imputation

Definition 5. (Pattern-Determining at time t_n) At time t_n the reference time series $\mathbf{R}_s$ *pattern-determine* time series s, written $\mathbf{R}_s, t_n \overset{\text{pd}}{\longrightarrow} s$, if for the k most similar anchor points $\mathbf{A}$ (cf. Def. 3) and patterns of length l the following holds for a small value ϵ:

$$\forall t_i, t_j \in \mathbf{A} : |s(t_i) - s(t_j)| \leq \epsilon$$

Example 9. In our running example (cf. Fig. 3), the query pattern $P(t_n{=}14{:}20)$ is based on the two reference time series $\mathbf{R}_s = \{r_1, r_2\}$. The $k = 2$ most similar anchor points are $\mathbf{A} = \{14{:}00, 13{:}35\}$, and the values of s at these two anchors are 21.9°C and 21.8°C, respectively. The two reference time series $\mathbf{R}_s$ pattern-determine s at time t_n (written $\mathbf{R}_s, 14{:}20 \overset{\text{pd}}{\longrightarrow} s$) with $\epsilon = |21.9°C - 21.8°C| = 0.1°C$. □

Pattern-determining time series guarantee that for a missing value $s(t_n)$ we find in the sliding window at least k similar patterns $P(t_{i_1}), \ldots, P(t_{i_k})$ to $P(t_n)$ and the missing value $s(t_n)$ is similar to the observed values $s(t_i)$.

Definition 6. (*Consistent Time Series*) Let s be a time series with missing value $s(t_n) = \textsc{Nil}$. Let $\hat{s}$ be a time series where the missing value $s(t_n)$ has been imputed, that is $\hat{s}(t_n) \neq \textsc{Nil}$ and $\forall t \in W \setminus \{t_n\} : \hat{s}(t) = s(t)$. Time series $\hat{s}$ is *consistent* if $\forall t \in \mathbf{A} : |\hat{s}(t) - \hat{s}(t_n)| \leq \epsilon$.

When TKCM imputes an incomplete time series s, we get an imputed time series $\hat{s}$. Intuitively, $\hat{s}$ is *consistent* if its value at the current time t_n is similar to past values when the reference time series observed a similar pattern.

LEMMA 5.2. *Let s be an incomplete time series with a missing value $s(t_n) = \textsc{Nil}$ and $\mathbf{R}_s$ its reference time series. Let $\hat{s}$ be the imputed time series produced by TKCM. If (a) $\mathbf{R}_s, t_n \overset{\text{pd}}{\longrightarrow} s$ and (b) $s(t_n)$ is imputed as defined in Eq. 4, $\hat{s}$ is a consistent time series.*

PROOF. Let $P(t_n)$ be the query pattern that TKCM constructs for the reference time series in $\mathbf{R}_s$ with pattern length l. TKCM looks for the k most similar anchor points $\mathbf{A}$ with respect to $P(t_n)$. Since the reference time series $\mathbf{R}_s$ pattern-determine s at time t_n, we have that $\forall t, t' \in \mathbf{A} : |s(t) - s(t')| \leq \epsilon$. The imputed value $\hat{s}(t_n)$ is the average of s at the anchor points (cf. Eq. 4). Since all values of s at $\mathbf{A}$ are similar among each other within a distance of ϵ, their mean $\hat{s}(t_n)$ is equally similar within an ϵ distance to all of them, i.e. $\forall t \in \mathbf{A} : |\hat{s}(t) - \hat{s}(t_n)| \leq \epsilon$. Consequently the imputed time series $\hat{s}$ is consistent. □

Next, we give an example of pattern-determining time series to illustrate what kind of phenomena TKCM can handle. Specifically, we show that sine waves of the form $f(t) = A \times \text{sind}(t\frac{360}{P} + \phi) + o$ with amplitude A, period P, offset o, and phase shift ϕ are pattern-determining and that TKCM achieves consistent imputation on these series. The experiments in Section 7 confirm that this also holds for real world time series.

LEMMA 5.3. *Assume $s(t) = A_1 \times \text{sind}(t\frac{360}{P} + \phi_1) + o_1$ and $r(t) = A_2 \times \text{sind}(t\frac{360}{P} + \phi_2) + o_2$. Then $\mathbf{R}_s = \{r\}$ is pattern-determining s at time t_n for $l > 1$, $k \geq 1$ and $L \geq kP + l$.*

PROOF. Observe that for $l > 1$, pattern $P(t_n)$ occurs exactly once every full period, i.e., $P(t) = P(t_n)$ only if $t = t_{n-iP}$ for every $i \in \mathbb{N}$. Since $L \geq kP + l$ we know there are k patterns $P(t)$, such that $P(t) = P(t_n)$. Since s has the same periodicity as r, we know that $\forall t, t' \in \mathbf{A} : |s(t) - s(t')| \leq 0 = \epsilon$. □

6. IMPLEMENTATION OF TKCM

6.1 Overview

To impute a missing value in a time series s at the current time t_n, TKCM performs three steps:

1. *Pattern Extraction:* Extract the anchor points of all candidate patterns from the streaming window W and compute the dissimilarity of these patterns to query pattern $P(t_n)$ (cf. Definition 2).

2. *Pattern Selection:* Select from the anchor points determined in step 1 the subset $\mathbf{A}$ of the time points that anchor the k most similar non-overlapping patterns (cf. Definition 3).

3. *Value Imputation:* Impute the missing value at t_n using anchor points $\mathbf{A}$ (cf. Definition 4).

In step 2, TKCM must find the k patterns that neither overlap each other nor $P(t_n)$, and that minimize the sum of dissimilarities with respect to $P(t_n)$. A simple greedy algorithm that sorts the anchor points according to dissimilarity and picks the first k ones that do not overlap fails to minimize the sum of dissimilarities. Therefore, we propose a dynamic programming scheme that exploits an optimal sub-structure of this problem. Let $D[j]$ denote the dissimilarity between the jth pattern in the window and the query pattern, and $M[i, j]$ denote the sum of dissimilarities of the i most similar non-overlapping patterns from among the first j patterns in W, where $i \leq j$. $M[i, j] = 0$ if $i = 0$, because no patterns have to be chosen. Similarly, $M[i, j] = \infty$ if $i > j$ because we cannot possibly find i non-overlapping patterns if we have only $j < i$ to choose from. Otherwise, we have two options: either (a) we omit the jth pattern and pick i patterns that possibly overlap it; or (b) we pick the jth pattern having dissimilarity $D[j]$ and have space left for $i{-}1$ patterns that do not overlap it. In the latter case, the first pattern to no longer overlap the jth pattern, if one exists, is the $(j - l)$th pattern.

This yields the following recurrence that minimizes the sum of dissimilarities:

$$M[i,j] = \begin{cases} 0 & \text{if } i = 0,\\ \infty & \text{if } i > j,\\ \min\begin{cases} M[i,j{-}1] \\ D[j] + M[i{-}1, j-l] \end{cases} & \text{otherwise} \end{cases} \tag{5}$$

The sum of dissimilarities of the k most similar anchor points is given by $M[k, L{-}2l{+}1]$, since $L{-}2l{+}1$ is the number of anchor points when we exclude the $l - 1$ first and l last time points in W (cf. Def. 3).

6.2 Algorithm

The implementation uses one ring buffer of length L for each time series s and an offset O into the ring buffers to efficiently update the streaming window. The value at time t_n is located at $s[O]$ and the oldest value at $s[(O{+}1)\%L]$, where % is the modulo operator. TKCM's pseudo code is listed in Algorithm 1. The input parameters are the ring buffer for the incomplete time series s with a missing value at $s[O]$, d ring buffers R for the reference time series, the window size L, the pattern length l, and the number of anchor points k. The algorithm stores the imputed value in $s[O]$ and returns it.

For processing, the algorithm uses an array D to store pattern dissimilarities, a $(k{+}1) \times (L{-}2l{+}2)$ matrix M to store the result for the dynamic programming algorithm, and an array A of size k to store the k most similar anchor points.

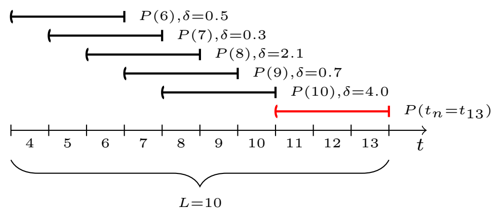
Lines 1–7 correspond to step 1, where the dissimilarities of all patterns in W are computed and stored in array D. The first $l{-}1$ and the last l time points are ignored as described above. In step 2, the algorithm finds the k most similar anchor points (Lines 8–23). Lines 8–14 implement the recurrence in Equation 5. $\max(j{-}l, 0)$ computes the predecessor of the jth pattern, yielding $j = 0$ if no such predecessor exists. Once matrix M is filled, $M[k, L{-}2l{+}1]$ contains the sum of dissimilarities of the k most similar anchor points $\mathbf{A}$. Finally, TKCM backtracks in lines 15–23 to find the anchor points $\mathbf{A}$. The algorithm starts in the lower-right most cell $M[k, L{-}2l{+}1]$ and applies the recurrence backwards. If for a cell $M[i, j]$ we have $M[i, j] = M[i, j{-}1]$ the jth anchor point is skipped as it is not part of the optimal solution, and the algorithm proceeds at cell $M[i, j{-}1]$. Otherwise, the jth

Algorithm 1: TKCM

```
Input: Ring buffer $s$, Array $R$ of $d$ ring buffers, window size $L$,
       pattern length $l$, and $k$.
Output: Imputed value in $s[O]$.
1  for $j \leftarrow 1$ to $L{-}2{*}l{+}1$ do
2      $D[j] \leftarrow 0$;
3      for $i \leftarrow 0$ to $d{-}1$ do
4          for $x \leftarrow 0$ to $l{-}1$ do
5              $y \leftarrow l{+}j{-}x{-}1$;
6              $D[j] \leftarrow D[j]{+}(R[i][(O{+}y)\%L]{-}R[i][(O{-}x)\%L])^2$;
7      $D[j] \leftarrow \text{sqrt}(D[j])$;
8  for $j \leftarrow 0$ to $L{-}2{*}l{+}1$ do
9      $M[0][j] \leftarrow 0$;
10     for $i \leftarrow 1$ to $k$ do
11         if $i > j$ then
12             $M[i][j] \leftarrow \infty$;
13         else
14             $M[i][j] \leftarrow \min(M[i][j{-}1], D[j]{+}M[i{-}1][\max(j{-}l, 0)])$;
15 $i \leftarrow k$;
16 $j \leftarrow L{-}2{*}l{+}1$;
17 while $i > 0$ do
18     if $M[i][j] = M[i][j{-}1]$ then
19         $j{-}{-}$;
20     else
21         $A[i{-}1] \leftarrow j$;
22         $i{-}{-}$;
23         $j \leftarrow \max(j{-}l, 0)$;
24 $s[O] \leftarrow 0$;
25 for $i \leftarrow 0$ to $k{-}1$ do
26     $s[O] \leftarrow s[O] + (s[(O{+}l{+}A[i]{-}1)\%L]/k)$;
27 return $s[O]$;
```

anchor point is added to $\mathbf{A}$ and the algorithm proceeds with cell $M[i{-}1, \max(j{-}l, 0)]$ until i reaches 0, indicating that k anchor points have been chosen. Finally, in lines 24–27, which correspond to step 3, the algorithm imputes the missing value using the k most similar anchor points according to Definition 4.

Example 10. The top of Fig. 8 shows a streaming window of length $L = 10$ with current time $t_n = t_{13}$. The query pattern of length $l = 3$ and all extracted patterns are shown as red and black intervals, respectively. The first (i.e., $j = 1$) pattern is $P(t_6)$ and the last pattern is $P(t_{10})$. The lower left table lists the patterns in the streaming window, their index j, predecessor, and dissimilarity with respect to query pattern $P(t_{13})$. For instance, $P(t_8)$ with index $j = 3$ has no non-overlapping predecessor, hence $\max(j - l, 0) = 0$. The right table shows the matrix M computed by Algorithm 1. For instance, $M[1, 1]$ is the result of computing $\min(D[1] + M[1{-}1, \max(1{-}3, 0)], M[1, 1{-}1]) = \min(0.5{+}0, \infty) = 0.5$. $M[2, 5] = 1.2$ in the lower-right corner contains the minimum sum of dissimilarity. To retrieve the k most similar non-overlapping patterns, the algorithm starts at $M[2, 5]$ and follows the highlighted path through the matrix: gray means that a pattern was omitted and green means that a pattern is part of the final result. For instance, $M[2, 5]$ is equal to $M[2, 4]$, hence the $j =$ 5th pattern $P(t_{10})$ is not part of the solution. $M[2, 4]$, in turn, is the result of $D[4]{+}M[1, 1] = 0.7{+}0.5 = 1.2$, hence $j =$ 4th pattern $P(t_9)$ is part of the solution. Continuing with $M[1, 1]$ we find another match before reaching $M[0, 0]$. The algorithm finds the $k = 2$ anchor points $\mathbf{A} = \{t_6, t_9\}$ and computes the imputed value $\hat{s}(t_{13}) = {}^1\!/\!_2(s(t_6) + s(t_9))$.



$P(t)$	j	$\max(j{-}l, 0)$	$D[j]$
$P(t_6)$	1	0	0.5
$P(t_7)$	2	0	0.3
$P(t_8)$	3	0	2.1
$P(t_9)$	4	1	0.7
$P(t_{10})$	5	2	4.0

	$j=0$	1	2	3	4	5
$i=0$	0	0	0	0	0	0
1	∞	0.5	0.3	0.3	0.3	0.3
2	∞	∞	∞	∞	1.2	1.2

Figure 8: Dynamic programming algorithm to compute the time points of the top $k = 2$ non-overlapping patterns of length $l = 3$ that minimize the sum of dissimilarities.

6.3 Complexity Analysis

LEMMA 6.1. *When the current time t_n advances, TKCM needs $O(1)$ time per stream s to update the corresponding ring buffer of size $O(L)$.*

PROOF. When t_n advances, a new value replaces an old value in the time series, requiring $O(1)$ time in a ring buffer of size L. □

LEMMA 6.2. *The time complexity of TKCM to impute a missing value is $O((l \times d + k) \times L)$.*

PROOF. Initially TKCM computes the dissimilarity of $O(L)$ patterns, each of size $l \times d$, having an overall time complexity of $O(l \times d \times L)$. Next the algorithm iterates over the $O(k \times L)$ sized dynamic programming matrix M. Hence the overall time complexity is $O(l \times d \times L + k \times L)$. □

LEMMA 6.3. *The space complexity of TKCM to impute a missing value is $O(k \times L)$.*

PROOF. The pattern extraction phase requires $O(L)$ space to store the dissimilarities of all patterns. TKCM needs $O(k \times L)$ space for matrix M. □

7. EXPERIMENTAL EVALUATION

In the experiments we simulate large blocks of consecutively missing values (e.g. one week). We repeatedly call TKCM to impute each missing value. This simulates a common sensor failure that requires a technician to reach a faulty weather station and replace the broken sensor. As accuracy measure we use the root mean square error (RMSE), defined as

$$\text{RMSE} = \sqrt{\frac{1}{|T|} \sum_{t_n \in T} (s(t_n) - \hat{s}(t_n))^2},$$

where T is the set of missing time points. The experiments are conducted on a Linux server, running Ubuntu 14.04 server edition. It is powered by an Intel Xeon X5650 CPU with a frequency of 2.67GHz and 24GB of main memory. TKCM is implemented in C and compiled with Clang 3.4-1, based on LLVM 3.4.

7.1 Datasets and Setup

We use both real-world and synthetic datasets in our experimental evaluation. First, we use the SBR dataset of meteorological time series in South Tyrol (cf. Sec. 1). Second, we shift the time series of the SBR data set by a (different) random amount up to one day and call this dataset SBR-1d. Third, we use the Flights dataset [3] that consists of eight time series, each of length 8801 (6 days). A time series describes at time t the number of airplanes that departed from a given airport and are in the air at time t. Fourth, we use the publicly available Chlorine dataset [1] used by SPIRIT [16]. This synthetic dataset was generated by a simulation of a drinking water distribution system; it describes the chlorine concentration at 166 junctions over a time frame of 4310 time points (15 days) with a sample rate of 5 minutes. The propagation of the chlorine level in the system causes phase shifts in the dataset. Fig. 9 shows an excerpt of three sample time series from each dataset. Each time series has different amplitudes, phase shifts, and trends.

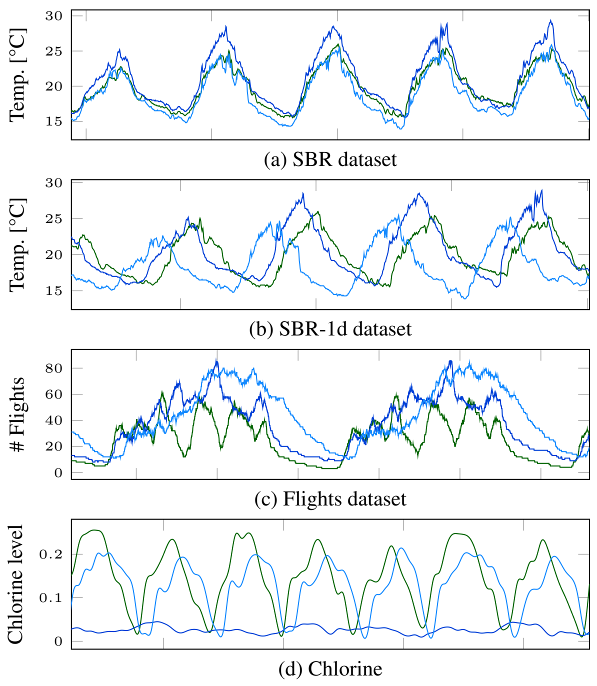


Figure 9: Three sample time series from each dataset.

We compare TKCM to three competitors: CD [13] (provided by the author), MUSCLES [25] (implemented in Matlab), and SPIRIT [23] (obtained from [1], Matlab code). SPIRIT's Matlab code does not impute missing values, hence we extended the code to use one autoregressive model per hidden variable as described in [23]. SPIRIT automatically adds or removes hidden variables as the streams evolve. When a hidden variable appears, a new autoregressive model of order $p = 6$ needs to be fitted, which requires at least p values of the new hidden variable before it can be used. If in that time a value needs to be imputed, the model is not yet ready. Consequently we fixed the number of hidden variables at two, which gave generally the best results in our experiments. For MUSCLES and SPIRIT we use a tracking window size of $p = 6$ as recommended by the authors [23, 25]. Contrary to the author's recommendation we set the exponential forgetting factor λ to 1 rather than to $0.96 \leq \lambda \leq 0.98$. We found that for $\lambda < 1$ the accuracy decreases, because the algorithms "forget" the old non-imputed (and accurate) values and adapt more to the new imputed (and inaccurate) values. CD has no parameters to tune. The code for our MUSCLES, SPIRIT, and TKCM implementations is available online[1].

[1]http://www.ifi.uzh.ch/en/dbtg/Staff/wellenzohn/dasa.html

7.2 Calibration

We begin with an initial calibration of TKCM's parameters d, k, and L. Unless otherwise noted we set the parameters to the following default values; $d = 3$ reference time series, $k = 5$ most similar anchor points, a streaming window of $L = 1$ year, and pattern length $l = 72$.

In the left column of Fig. 10 we show TKCM's accuracy for increasing values of d on three datasets; for brevity we omit the SBR dataset as it shows identical behavior to the SBR-1d dataset. TKCM's accuracy significantly increases (that is the RMSE decreases) as d increases up to $d = 3$ reference time series, while $d > 3$ does not provide significantly better accuracy. Since the Flights dataset has only 8 time series, we can set d at most to 7. The right column of Fig. 10 shows the impact of parameter k on TKCM's accuracy. In general we tend to pick small values of k, e.g., $k \in [3, 10]$ to get the best possible (most similar) patterns from the window. Larger values of k may add less similar patterns, in particular for short streaming windows. We observed that for the two small datasets Flights and Chlorine $k = 5$ is sufficient. TKCM's accuracy noticeably decreases on the Flights dataset for $k > 5$, because the dataset contains only measurements for 6 days and if one day is missing we try to find more than 5 similar situations within 5 days, which makes no sense. For the larger (1 year) SBR and SBR-1d datasets we found that there is a marginal accuracy increase even from $k = 5$ to $k = 10$, after which the accuracy remains stable. Therefore, our recommendation is to use a small value of k, e.g. $k = 5$, and if the dataset is large, one can safely double k.

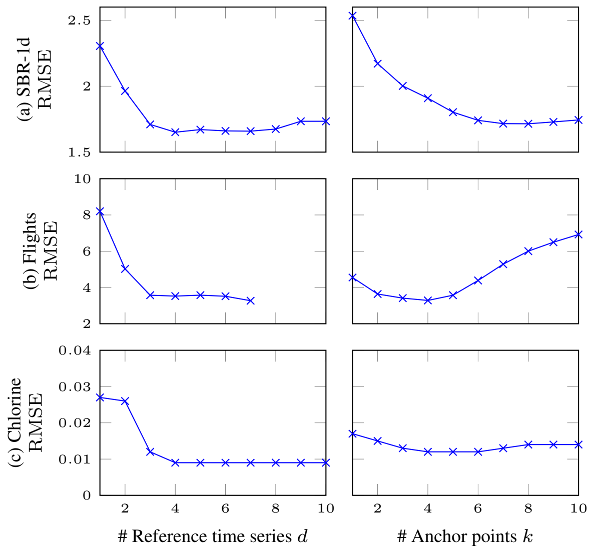


Figure 10: Calibration shows that $d = 3$ and $k = 5$ are good default values.

For the small Flights and Chlorine datasets (6 and 15 days, respectively) we use in our experiments the entire time range as streaming window length L. For the SBR and SBR-1d datasets we use a streaming window length $L = 105120$ (1 year), because one year covers the whole temperature range and contains each pattern several times. This choice is conservative; we found that already a window size of 6 months gives a good accuracy of RMSE = 1.8°C, which only dropped to 1.7°C for a 5 year window. In general, larger window sizes L provide only a slightly superior accuracy.

7.3 Accuracy

7.3.1 Pattern Length l

For shifted time series, the pattern length l is the key to an accurate and robust imputation. In Fig. 11, we evaluate l by varying the pattern length l from 1 to 144 (i.e., a pattern that spans 12 hours). As expected, for the (non-shifted) SBR dataset l has close to no impact on TKCM's accuracy, because there is a high linear correlation between the incomplete time series and its $d = 3$ reference time series. For the SBR-1d dataset the RMSE drops by about 0.5°C (25%) by increasing l to 72. On the flight dataset we observe an improvement of 50% for $l = 72$ and 60% for $l = 144$. The reason why we see an improvement beyond $l = 72$ is the different sample rate of 1 minute of the Flights dataset as opposed to the 5 minutes in the SBR dataset. While $l = 72$ yields a pattern that spans 6 hours in the SBR dataset, it only spans 1 hour in the Flights dataset. On the Chlorine dataset we observe an accuracy increase of 60% with pattern length $l = 72$, after which the accuracy slightly decreases.

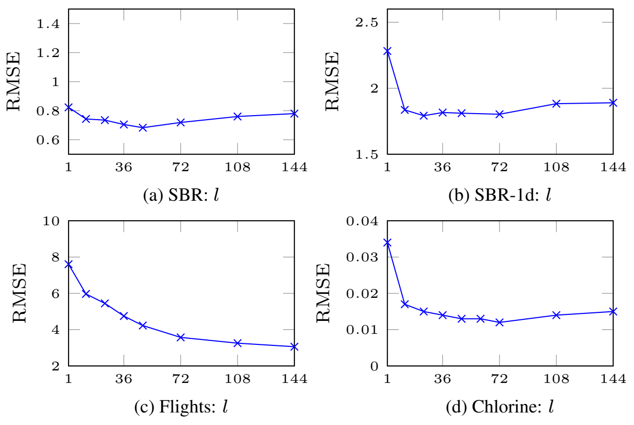


Figure 11: Pattern length l evaluated on each dataset.

To put these raw numbers into perspective and see the real impact of l, we compare TKCM's recovery of an incomplete time series s in Fig. 12 with pattern length $l = 1$ (left column) and $l = 72$ (right column). Observe how much TKCM's recovery oscillates with $l = 1$ and how well TKCM adapts to the assumed missing time series s with $l = 72$. Even for the SBR dataset there is a slight oscillation, albeit minimal compared to the three shifted datasets. The reason for this strong oscillation when $l = 1$ is that with a short pattern, the reference time series do not pattern-determine the incomplete time series. The difference in dissimilarity between "good" patterns and "bad" patterns is too small for TKCM to detect, as explained in Sec. 5.1. Put differently, in the presence of shifts, time series are no longer linearly correlated; whenever a reference time series observes a very similar value, the incomplete time series has very different values.

Fig. 13a shows the scatterplot of an incomplete time series s in the Chlorine dataset against one of its reference time series r_1. There is clearly no strong linear correlation ($\rho(s, r_1) = 0.5$): e.g. for $r_1(t) = 0.1$, $s(t)$ has two different values (0 and 0.15). Fig. 13b shows that the average ϵ (cf. Def. 5) decreases as l increases on the Chlorine dataset with $k = 5$. Value $\epsilon = \max_{t,t' \in \mathbf{A}} |s(t) - s(t')|$ essentially describes the range of the values of s at the k most similar anchor points $\mathbf{A}$. The lower ϵ gets, the less the values of s differ at the most similar anchor points, which indicates that the reference time series strongly pattern determine s for k and l. Notice that the average ϵ increases after $l = 72$, which

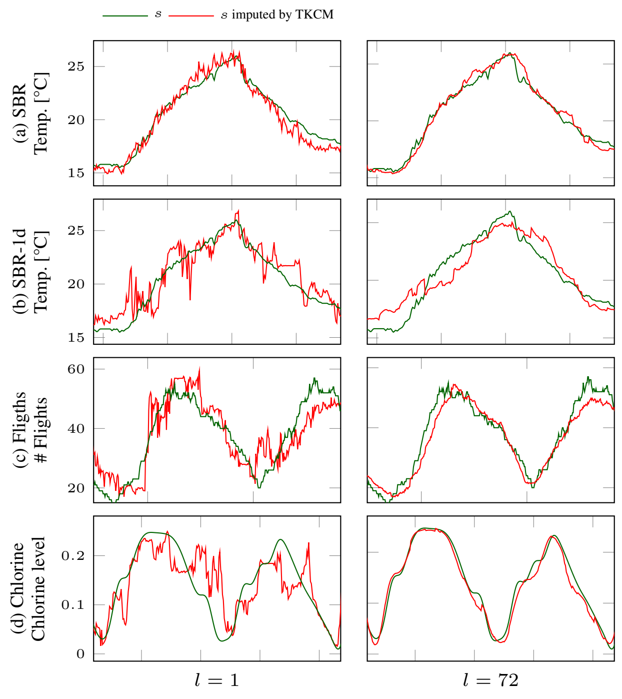


Figure 12: The reason for the strong oscillation in TKCM's recovery with $l = 1$ are shifts in the reference time series. Increasing the pattern length l helps TKCM to detect shifts.

coincides with our observations in Fig. 11d.

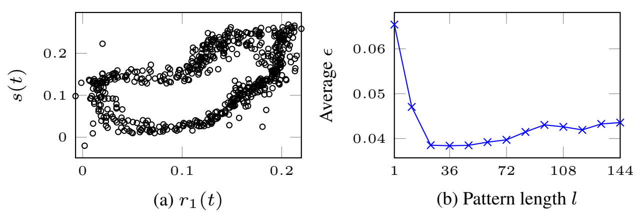


Figure 13: *Left:* Scatterplot of s against the reference time series r_1. *Right:* Range of s at the k anchor points (Chlorine dataset).

7.3.2 *Missing Block Length*

In Fig. 14 we study TKCM's accuracy in terms of the length of the missing block, i.e., the number of consecutively missing values that TKCM needs to impute. First, we use our large dataset SBR-1d and simulate sensor failures of up to several weeks. Fig. 14a shows that the accuracy of TKCM only slightly decreases by 0.2°C as the block length grows from 1 to 4 weeks, after which the accuracy plateaus. Next, we increase the missing block length for the small dataset Chlorine from 10% to 80% of the dataset size. We start with the remaining 90% to 20% of the dataset in the streaming window of length $L = 4310$ and impute the rest of the dataset as missing values. Fig. 14b shows that also for this case the accuracy decreases only slowly.

7.3.3 *Comparison with Competitors*

SBR. We first perform a baseline comparison of all algorithms on the SBR dataset that has no phase shifts. Fig. 15a shows an excerpt

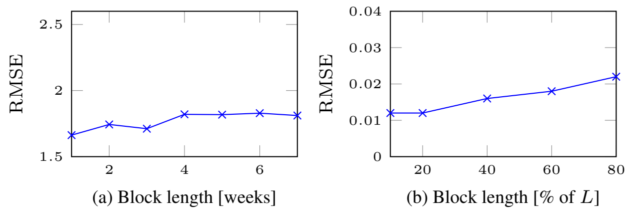


Figure 14: Impact of the missing block length on the accuracy (Chlorine dataset).

of the experiment where we assume a block of values is missing in time series s, and impute the missing values with each approach. TKCM, SPIRIT, MUSCLES, and CD perform virtually equally well. Observe that the last valley in the temperature curve shows a higher temperature than the previous valleys. While TKCM and CD are able to capture this trend, MUSCLES and SPIRIT impute a too low value. The most likely reason is that the models that MUSCLES and SPIRIT build are not able to adapt quickly enough to the new behavior of the time series, while TKCM adapts instantaneously to the changing behavior.

SBR-1d. Next we impute the same block of missing values in the SBR-1d dataset that has shifted time series. Observe how TKCM's accuracy only slightly decreases in Fig. 15b; TKCM slightly misses the second last downwards slope, but the last valley is again accurately imputed. SPIRIT completely misses the amplitude; its recovered peaks are too low and its valleys are too high in temperature. Moreover, the overall trend of the missing block is not well recovered. MUSCLES recovery is borderline at best; the overall periodicity of the signal is recovered, but MUSCLES was not able to recover any feature of s. Moreover, s has a slightly increasing temperature trend, but MUSCLES' recovery has a decreasing trend. CD's recovery is shifted with respect to s, as also indicated by the discontinuous imputation at the very beginning of the missing block.

Flights. On the Flights dataset we observe a similar behavior as in the previous experiment. Fig. 15c shows that TKCM captures each peak and valley accurately, while SPIRIT's accuracy decreases over time. Initially, the trend of s is vaguely captured, but after the highest peak, the trend of SPIRIT's imputation is *inverse* (i.e., peaks and valleys are swapped) with respect to the true signal. Again, MUSCLES produces an extremely smoothed signal, that does not resemble the true time series; peaks and valleys are not recovered. CD's recovery is only partially shown as many recovered values are *negative*. Most likely, the large block of missing values (ca. 20% of the dataset) is the reason.

Chlorine. In the Chlorine dataset TKCM captures the trend of s generally well, the valleys are almost perfectly recovered, while the peaks are slightly less accurate. SPIRIT's recovery does not capture the amplitude of s, and also the trend of the recovery does not match that of s. MUSCLES completely misses the first peak, imputing it with a valley instead; also the general trend of MUSCLES' recovery does not resemble s. In this dataset we found MUSCLES and SPIRIT to perform with widely differing accuracies, sometimes their imputations is good, sometimes worse than in this example. There is no clear pattern when either approach works or fails. CD's recovered signal has a very small amplitude and also the trend does not capture that of s.

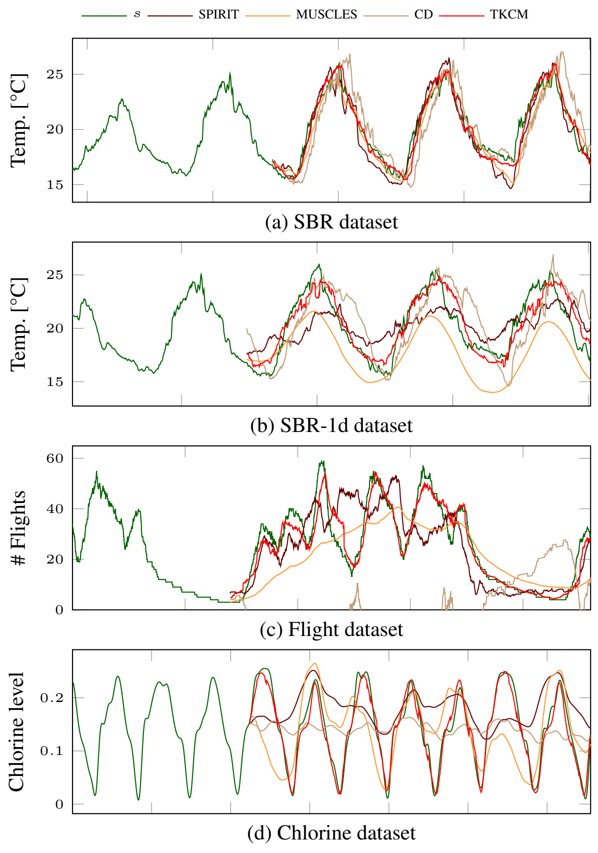


Figure 15: Imputation of incomplete time series s with different imputation techniques on four different datasets.

Summary. Fig. 16 shows the RMSE for all compared algorithms on each dataset. In this comparison we impute 4 time series per data set; with missing block lengths per time series of 1 week in the SBR and SBR-1d datasets, and 20% of the dataset size for Flights and Chlorine. All algorithms are given the same amount of data (L measurements per time series). We use $L = 6$ months for the SBR and SBR-1d datasets, because of CD's prohibitively large runtime for $L = 1$ year (our default) and the default L for the remaining datasets. The experiments show that only for the non-shifted SBR dataset all algorithms provide a comparable accuracy. For the remaining three shifted datasets, TKCM clearly outperforms its competitors both in terms of perceived accuracy (Fig. 15) and raw RMSE (Fig. 16). Our general observation is that non-shifted linearly-correlated data poses no significant challenge to any algorithm. As soon as shifts are present in the data, the accuracy of state-of-the-art solutions is largely unpredictable, ranging from good to unusable.

7.4 Runtime

As discussed in Sec. 6.3, TKCM's time complexity is linear with respect to all parameters (l, d, k, and L) as confirmed by Fig. 17. In this experiments on the SBR-1d dataset we vary each parameter, leaving the other three parameters at their defaults ($l = 72$, $d = 3$, $k = 5$ and $L = 1$ year). Fig. 17a and Fig. 17b show TKCM's runtime with respect to the size of the query pattern $P(t_n)$, Fig. 17c shows the impact of the number of anchor points k, and Fig. 17d shows the impact of the streaming window size L on the runtime. Parameter L has the largest impact on TKCM's runtime, followed

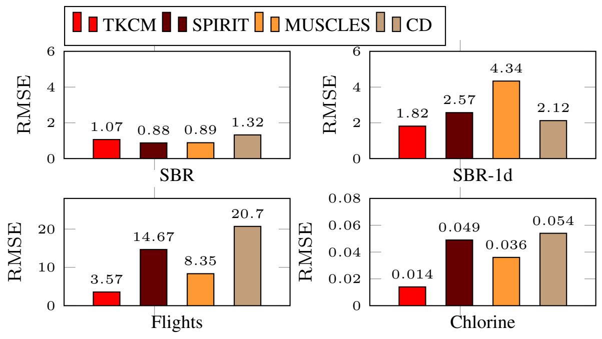


Figure 16: Comparison of TKCM, SPIRIT, MUSCLES, and CD for each dataset.

by l and d with similar impact. Parameter k is relatively cheap – even if set to very large values, e.g., $k > 50$. For our default parameter settings we observe a runtime of approximately 2 seconds to impute a single missing value.

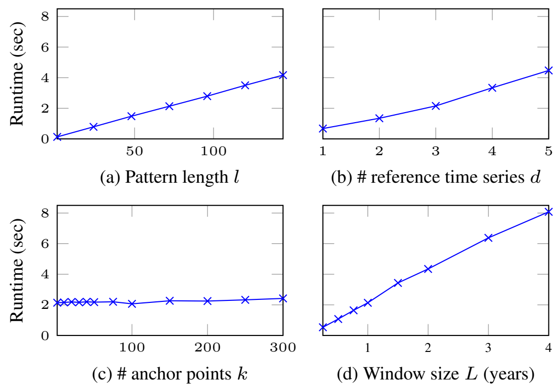


Figure 17: Runtime experiments. TKCM's time complexity is linear with respect to all its parameters l, d, k, and L (SBR-1d dataset).

Performance breakdown. As described in Sec. 6 the two main phases of TKCM are *pattern extraction* (PE) and *pattern selection* (PS). In our default setup, the PE-phase accounts for 92% of TKCM's overall runtime. If we further subdivide the PE-phase, we see that 82% of the overall runtime are required to fetch data from main memory and 10% are used to compute the pattern dissimilarity δ. If we increase k to 300 we see the runtime of the PS-phase climbing from 8% up to 25%. Thus, for the default value of k, the runtime incurred by the PS-phase is outweighed by the PE-phase. Hence, to improve TKCM's performance, future research must focus on speeding up the pattern extraction phase.

Comparison. A direct comparison of the runtimes of the considered approaches is not meaningful, because the systems are implemented in different programming languages (TKCM in C, CD in Java, MUSCLES and SPIRIT in Matlab). To give a rough feeling for the overall performance we consider each approach in turn. CD is an offline algorithm and not applicable to streams. CD's matrix decomposition lasted in our experiments roughly 20 minutes per execution and is hence not applicable to streaming environments. Both SPIRIT and MUSCLES required one millisecond to impute one missing value, TKCM requires roughly 2 seconds.

8. CONCLUSION AND FUTURE WORK

We studied the problem of missing values in meteorological streams of time series data and presented an algorithm, termed TKCM, to accurately impute missing values in a streaming environment. If the current value in a time series s is missing, TKCM determines a two-dimensional query pattern over the last l measurements of d reference time series. It then retrieves the anchor points of the k most similar non-overlapping patterns to the query pattern. The missing value is computed from the values of s at these k anchor points. We show that TKCM achieves consistent imputation if the reference time series pattern-determine s, which covers non-linear relationships between time series such as phase shifts. An extensive experimental evaluation using four real-world and synthetic datasets confirms that TKCM is accurate and outperforms state-of-the-art competitors.

Future work points in several directions. First, we will work on the efficiency of TKCM, in particular the pattern extraction phase, which proved to be the most time-consuming component. In particular, we plan to reduce the number of extracted patterns by pruning patterns that cannot possibly belong to an optimal solution. Second, we plan to investigate how to automatically determine the best candidate reference time series, although in many application domains (and especially in meteorology) we can rely on human experts. Third, we plan to compare different dissimilarity functions δ (e.g. L_1-norm, DTW [9], etc.). Moreover, it would be interesting to compute an alignment between shifted time series (e.g., using DTW [9]) and to compare TKCM's accuracy on the aligned time series using a pattern length $l = 1$ to the accuracy on the shifted time series using $l > 1$.

9. ACKNOWLEDGMENTS

The work has been done as part of the DASA project, which is funded by the Foundation of the Free University of Bozen-Bolzano. We wish to thank our partners at the Südtiroler Beratungsring and the Research Centre for Agriculture and Forestry Laimburg for the good collaboration and helpful domain insights they provided, in particular Armin Hofer, Martin Thalheimer, and Robert Wiedmer. We also want to thank Mourad Khayati for his input and for sharing his CD implementation used in the experimental evaluation. We thank the anonymous reviewers for their valuable comments and suggestions.

10. REFERENCES

[1] SPIRIT project. https://www.cs.cmu.edu/afs/cs/project/spirit-1/www/. Accessed: 2016-05-29.

[2] G. E. A. P. A. Batista and M. C. Monard. An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence*, 17(5-6), 2003.

[3] A. Behrend and G. Schüller. A case study in optimizing continuous queries using the magic update technique. In *SSDBM*, pages 31:1–31:4, 2014.

[4] G. E. P. Box and G. Jenkins. *Time Series Analysis, Forecasting and Control*. Holden-Day, Incorporated, 1990.

[5] B. Chiu, E. Keogh, and S. Lonardi. Probabilistic discovery of time series motifs. In *KDD*, pages 493–498, 2003.

[6] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. *SIGMOD Rec.*, 23(2):419–429, May 1994.

[7] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems*, 3(3):263–286, 2001.

[8] E. Keogh, J. Lin, and A. Fu. HOT SAX: Efficiently finding the most unusual time series subsequence. In *ICDM*, pages 226–233, 2005.

[9] E. J. Keogh. Exact indexing of dynamic time warping. In *VLDB*, 2002.

[10] E. J. Keogh and T. Rakthanmanon. Fast shapelets: A scalable algorithm for discovering time series shapelets. In *ICDM*, pages 668–676, 2013.

[11] M. Khayati and M. H. Böhlen. REBOM: recovery of blocks of missing values in time series. In *COMAD*, pages 44–55, 2012.

[12] M. Khayati, M. H. Böhlen, and J. Gamper. Memory-efficient centroid decomposition for long time series. In *ICDE*, pages 100–111, 2014.

[13] M. Khayati, P. Cudré-Mauroux, and M. H. Böhlen. Using lowly correlated time series to recover missing values in time series: a comparison between SVD and CD. In *SSTD*, pages 237–254, 2015.

[14] L. Li, J. McCann, N. S. Pollard, and C. Faloutsos. DynaMMo: Mining and summarization of coevolving sequences with missing values. In *KDD*, pages 507–516, 2009.

[15] P. Merlin, A. Sorjamaa, B. Maillet, and A. Lendasse. X-SOM and L-SOM: A double classification approach for missing value imputation. *Neurocomputing*, 73(7-9):1103–1108, 2010.

[16] S. Papadimitriou, J. Sun, and C. Faloutsos. Streaming pattern discovery in multiple time-series. In *VLDB*, pages 697–708, 2005.

[17] S. Papadimitriou, J. Sun, C. Faloutsos, and P. S. Yu. Dimensionality reduction and filtering on time series sensor streams. In *Managing and Mining Sensor Data*, pages 103–141. 2013.

[18] J. L. H. Paulhus and M. A. Kohler. Interpolation of Missing Precipitation Records. *Monthly Weather Review*, 80(8), Aug. 1952.

[19] D. Rubin. Multiple Imputation after 18+ Years. *Journal of the American Statistical Association*, 91(434), 1996.

[20] J. L. Schafer and J. W. Graham. Missing data: our view of the state of the art. *Psychological Methods*, 7, 2002.

[21] J. Shieh and E. Keogh. iSAX: Indexing and mining terabyte sized time series. In *KDD*, pages 623–631, 2008.

[22] A. Sorjamaa, P. Merlin, B. Maillet, and A. Lendasse. SOM+EOF for finding missing values. In *ESANN*, pages 115–120, 2007.

[23] J. Sun, S. Papadimitriou, and C. Faloutsos. Online latent variable detection in sensor networks. In *ICDE*, pages 1126–1127, 2005.

[24] O. G. Troyanskaya, M. N. Cantor, G. Sherlock, P. O. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman. Missing value estimation methods for DNA microarrays. *Bioinformatics*, 17(6), 2001.

[25] B. Yi, N. Sidiropoulos, T. Johnson, H. V. Jagadish, C. Faloutsos, and A. Biliris. Online data mining for co-evolving time sequences. In *ICDE*, pages 13–22, 2000.

[26] C. Yozgatligil, S. Aslan, C. Iyigun, and I. Batmaz. Comparison of missing value imputation methods in time series: the case of turkish meteorological data. *Theoretical and Applied Climatology*, 112(1-2), 2013.