REBOM: Recovery of Blocks of Missing Values in Time Series

Mourad Khayati
Department of Informatics
University of Zürich
Binzmühlestrasse 14, CH-8050
Zürich, Switzerland
mkhayati@ifi.uzh.ch

Michael H. Böhlen
Department of Informatics
University of Zürich
Binzmühlestrasse 14, CH-8050
Zürich, Switzerland
boehlen@ifi.uzh.ch

ABSTRACT

The recovery of blocks of missing values in regular time series has been addressed by model-based techniques. Such techniques are not suitable to recover blocks of missing values in irregular time series and restore peaks and valley. We propose REBOM (REcovery of BlOcks of Missing values): a new technique that reconstructs shapes, amplitudes and width of missing peaks and valleys in irregular time series. REBOM successfully reconstructs peaks and valleys by iteratively considering the time series itself and its correlation to multiple other time series. We provide an iterative algorithm to recover blocks of missing values and analytically investigate its monotonicity and termination. Our experiments with synthetic and real world hydrological data confirm that for the recovery of blocks of missing values in irregular time series REBOM is more accurate than existing methods.

Keywords

Missing blocks recovery, irregular time series, Singular Value Decomposition, ranking matrix.

1. INTRODUCTION

Time series data arise in a variety of domains, such as environmental, telecommunication, financial, and medical data. For example, in the field of hydrology, sensors are used to capture environmental phenomena including temperature, air pressure, and humidity at different points in time. For such data, it is not uncommon that more than 20% of the data is missing as blocks, i.e., multiple consecutive measurements are missing.

Existing techniques effectively recover blocks of missing values in regular time series, i.e., time series series containing peaks and valleys with a possibly varying frequency or amplitude that follow one or more periodic models, e.g., the sinus model where the frequency varies over time. The recovery accuracy of these techniques decreases for irregular time series, i.e., time series containing peaks and valleys that do not follow any model. In this work, we address the problem of finding the optimal recovery of blocks of missing values in irregular time series. We propose REBOM (REcovery of BlOcks of Missing values), a new data driven recovery technique for blocks of missing values that is able to restore missing peaks and valleys. We use the correlation [1] between time series to recover blocks of missing values. Intuitively, time series that tend to change their peaks and valleys simultaneously are correlated and we use the Pearson coefficient to quantify this correlation.

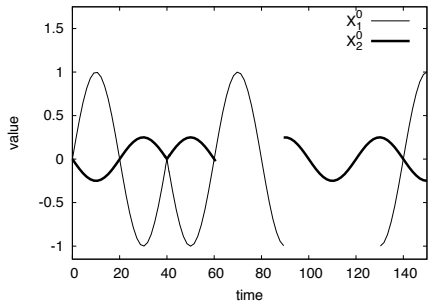
REBOM is an iterated low rank Singular Value Decomposition (SVD) [2]. We decompose a matrix $\mathbf{V}$ of correlated time series, where missing values have been initialized through linear interpolation combined with nearest neighbor imputation, into the product $\mathbf{L} \times \mathbf{\Sigma} \times \mathbf{R}^T$ of three matrices. By nullifying the smallest singular value of $\mathbf{\Sigma}$ we give higher priority to the correlation between the time series. The subsequent matrix multiplication yields an approximation of $\mathbf{V}$ that better approximates the missing values. After each iteration, the ranking of the most correlated time series with respect to the time series to recover, is updated. The iterative recovery terminates if the total ranking, which is determined by considering all observations of the time series, is identical to the partial ranking, which is determined by considering only observations with timestamps of missing values. If the total and the partial ranking are equal, the correlation can no longer be used to improve the recovery of missing values.

Problem definition: Assume a set of n irregular correlated time series $\mathbf{X}^0 = \{X_1^0, X_2^0, \ldots, X_n^0\}$ where $X_1^0, X_2^0, \ldots, X_n^0$ contain blocks of missing values. We propose a recovery method that determines, in j iterations, a set of time series $\widetilde{\mathbf{X}}^j = \{\widetilde{X}_1^j, \widetilde{X}_2^j, \ldots, \widetilde{X}_n^j\}$ where the missing blocks of $X_1^0, X_2^0, \ldots, X_n^0$ have been restored.

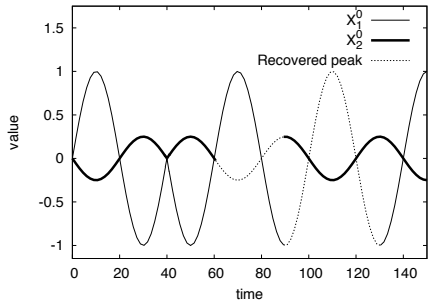
The result of REBOM for the recovery of peaks and valleys for two correlated time series is illustrated in Figure 1. Each time series is displayed as a 2d plot where the x-axis shows the timestamp t and the y-axis the value v for a given t. X_1^0 represents an air pressure time series and contains a missing block for the time range $]90, 130[$. X_2^0 represents a temperature time series that contains a missing block for the time range $]60, 90[$. REBOM can be used to restore the missing blocks of X_1^0 and X_2^0.

Figure 1 illustrates that REBOM accurately recovers *shape*, *amplitude* and *width* of the missing blocks. REBOM

The 18th International Conference on Management of Data (COMAD), 14th-16th Dec, 2012 at Pune, India.



(a) Original Time Series



(b) Restoration of Missing Blocks of X_1^0 and X_2^0

Figure 1: Recovery Performed by REBOM

detects that the peaks and valleys of X_1^0 and X_2^0 are correlated (high pressure corresponds to low temperature and vice versa). The shape and the width of the missing block are recovered from the position of the local extrema of X_1^0 with respect to the local extrema of the correlated time series X_2^0. The amplitude of the missing block of X_1^0 is recovered based on the two preceding peaks of X_1^0.

At the technical level, we show how to iterate the low rank SVD and we analytically investigate the main properties of the method. The main contributions of this paper are:

- We propose REBOM: an iterated low rank SVD that iteratively refines the initial recovery of missing values.
- We propose a greedy algorithm that repeatedly selects a time series with missing values that have been initialized and uses the k most correlated time series to iteratively refine the recovery of the missing values.
- We prove that our greedy algorithm is stepwise monotonic, i.e., the accuracy of the recovery increases by choosing, at each step, the most correlated time series. The algorithm terminates when the set of the most correlated time series does not change anymore.
- We empirically show that the recovery accuracy of REBOM is invariant to the initial recovery. Different initialization methods lead to the same recovery accuracy but with different number of iterations.
- We present an experimental evaluation of the accuracy of our technique that compares REBOM to state-of-the-art techniques for the recovery of blocks of missing values. The results show the superiority of our algorithm for the restoration of peaks and valleys.

The rest of the paper is organized as follows. Section 2 reviews related work on reduction methods and existing techniques for imputing missing values. Section 3 defines the initialization method and describes the basics of the low rank SVD. Section 4 introduces and discusses REBOM and its properties. Section 5 empirically compares the results of REBOM to other techniques proposed in the literature for the recovery of blocks of missing values.

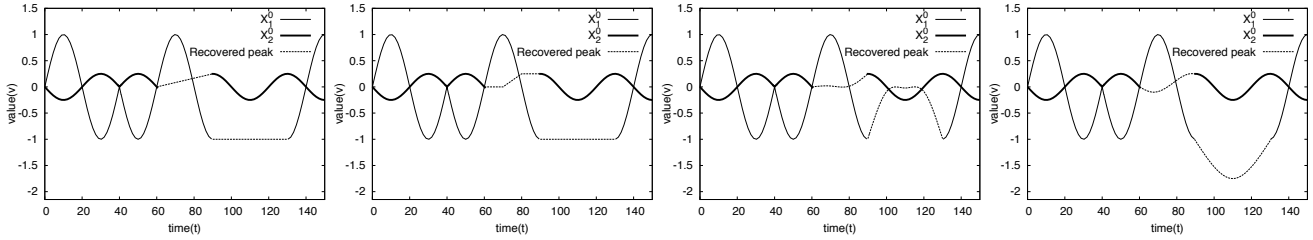
2. RELATED WORK

Prediction models such as Maximum Likelihood Estimation (MLE) [3], Bayesian Networks (BN) [4, 5] and Expectation Maximization (EM) [6] were used to estimate single missing values or small blocks of missing values in time series. These techniques are parametric and require a specific type of data distribution, e.g, Gaussian distribution. Therefore, they only perform well for the recovery of blocks of missing values in regular time series where peaks and valleys follow a periodic model of constant frequency and amplitude.

Li et al. [7] presented an approach called DynaMMo that is based on Expectation Maximization (EM) and Kalman Filter [8]. This technique is intended to recover missing blocks in non linear time series that contain peaks and valleys. DynaMMo allows to use one reference time series in addition to the time series that contains the missing block. The Kalman Filter uses the data of the time series that contains missing blocks together with a reference time series, to estimate the current state of the missing blocks. This estimation is performed as a multi step process that uses two different estimators. The first estimator represents the current state and the second estimator represents the initial state and the error of the estimation. For every step of the process, an EM method predicts the value of the current state and then the two estimators are used to refine the predicted values of the current state and to maximize their likelihood. DynaMMo does not allow to use more than one reference time series for the block recovery. DynaMMO performs an accurate block recovery for any type of regular time series. The accuracy of the block recovery decreases for irregular time series (cf. Section 5).

Techniques that rely on basic statistical methods such as mean imputation, piecewise approximation (linear spline, cubic spline, ...) [9, 10], regression [11, 12] and k Nearest Neighbors [13, 14] have been proposed for the recovery of blocks of missing values. Figures 2(a) and 2(b) illustrate the block recovery performed respectively by linear spline and k nearest neighbor using values at t=60 and t=90. Figure 2(c) shows that the regression method replaces missing values by points lying on the line that minimizes the regression error of all existing points. These techniques are not able to accurately recover any of the two missing blocks in X_1^0 and X_2^0. The cubic spline technique finds a third order polynomial that connects three successive values. Figure 2(d) shows that the cubic spline replaces the missing block by a block opposite to the one that precedes the missing block. Cubic spline is able to perform a good recovery only for the missing block of X_1^0. All basic methods are not suitable techniques for block recovery in regular time series where peaks and valleys follow a periodic model of varying amplitude or frequency, or in irregular time series.

Kurucz et al. [15] proposed a technique based on EM and Singular Value Decomposition (SVD) [16, 17, 18, 19] for



(a) Linear Spline Recovery (b) k Nearest Neighbor Recovery (c) Regression Recovery (d) Cubic Spline Recovery

Figure 2: Recovery using Different Techniques

comparing recommender systems where one of them contains missing values. A recovery of the missing values is performed before the comparison process. Each recommender system is represented by one column of values in a rating matrix which is decomposed using SVD. The result of the decomposition is modified using a method called gradient boosting [20]. The EM algorithm is then applied to refine the result of gradient boosting. The proposed solution dynamically discovers data dependencies from coordinate axes that represent the recommender systems and is applicable for more than one reference recommender system. However, the application of gradient boosting on different recommender systems looses the dependencies among the original values of recommender systems. Therefore, this technique yields bad results for block recovery in case where more than one recommender system contains missing blocks.

Tree-based methods were proposed to impute missing values. He [21] and Ding and Simonoff [22] present an overview of tree classification methods that are able to replace missing values in time series. These trees find the optimal way to classify missing values using a regression approach and are called Classification and Regression Trees (CART). These techniques are designed to create a classification of the missing values. Missing values that belong to the same class will be recovered with the same value. Therefore, these methods are not able to effectively restore missing peaks and valleys in regular and irregular time series.

3. PRELIMINARIES AND BACKGROUND

3.1 Notation

We use the following notation: sets and vectors are upper-case, matrices are upper-case bold, and elements of sets and matrices are lower-case. A time series $X_1 = \{x_1, x_2, \ldots, x_n\}$ is a set of n observations. Each observation x_j from X_1 is a pair (t_j, v_j) where t_j and v_j are respectively the timestamp and the value of the observation. $T_1 = \{t \mid (t, \_) \in X_1)\}$ denotes the set of all timestamps from X_1; $V_1 = \{v \mid (\_, v) \in X_1)\}$ denotes the vector of all values from the time series X_1. A time series X_1 with missing values that have not been recovered yet, is denoted as X_1^0.

3.2 Preprocessing of Time Series

The first preprocessing step uses basic statistical methods to initialize all missing values. After the initialization the timestamps of all time series are aligned.

DEFINITION 1 (MISSING TIMESTAMPS). *Given a set of n time series $\{X_1^0, \ldots, X_n^0\}$, the set of missing timestamps of time series X_i^0 with respect to the timestamps of the other time series is $T_i^0 = \{t \mid ((t, \_) \in X_1^0 \vee \ldots \vee (t, \_) \in X_n^0) \wedge (t, \_) \notin X_i^0\}$.*

Note that missing timestamps of one time series have to be present in at least another time series. Timestamps missing in all time series are not considered. An additional preprocessing step can be added if such timestamps shall be recovered as well.

$X_1^1 = \{(t_1, v_1), ..., (t_n, v_n)\}$ is the initial recovery of X_1^0 iff $\forall i \in \{1, \ldots, n\}$

$$(t_i, v_i) = \begin{cases} (t_i, v_i) \textbf{ if } (t_i, v_i) \in X_1^0 \\ \textbf{Else} \begin{cases} (t_i, v) \textbf{ if } (s(t_i), \_) \notin X_1^0, \\ \qquad\qquad (p(t_i), v) \in X_1^0 \\ (t_i, v) \textbf{ if } (p(t_i), \_) \notin X_1^0, \\ \qquad\qquad (s(t_i), v) \in X_1^0 \\ (t_i, \frac{(t_i - p(t_i))(s(v_i) - p(v_i))}{s(t_i) - p(t_i)} + s(v_i)) \\ \qquad\qquad \textbf{otherwise} \end{cases} \end{cases}$$

$p(t_i) = max\{t_j \mid (t_j, \_) \in X_1^0 \wedge t_j < t_i\}$ is the predecessor of timestamp t_i in X_1^0 and $s(t_i) = min\{t_j \mid (t_j, \_) \in X_1^0 \wedge t_j > t_i\}$ is the successor timestamp of t_i in X_1^0. Similarly, $p(v_i) = \{v_j \mid (t_j, \_) \in X_1^0 \wedge t_j = p(t_i)\}$ is the predecessor of value v_i in X_1^0 and $s(v_i) = \{t_j \mid (t_j, \_) \in X_1^0 \wedge t_j = s(t_i)\}$ is the successor value of v_i in X_1^0. Thus, the initial recovery of the missing values is a linear interpolation. If the missing values occur as the first or the last elements of X_1^0, we use the nearest neighbor imputation.

Two time series X_1^1 and X_2^1 with initialized missing values define a set of multidimensional points: $\{(v, v') \mid (t, v) \in X_1 \wedge (t, v') \in X_2\}$. The second preprocessing step constructs a matrix with n m-dimensional points from m time series with n observation each.

EXAMPLE 1. *Figure 3 shows two time series X_1^0 and X_2^0 with missing values, the initialized time series X_1^1 and X_2^1, and the set of multidimensional points $\mathbf{V}$. The initialized missing values are highlighted in gray.*

From Definition 1 we get $T_1^0 = \{100, 110, 120\}$ and $T_2^0 = \{70, 80\}$.

3.3 Low Rank Matrix Decomposition

3.3.1 Singular Value Decomposition

The Singular Value Decomposition (SVD) is a matrix decomposition method that decomposes a matrix $\mathbf{V}$ into three matrices $\mathbf{L}$, $\mathbf{\Sigma}$ and $\mathbf{R}^T$. The product of the three matrices is equal to $\mathbf{V}$.

$\mathbf{X}_1^0$	
t	v
0	0
10	1
20	0
30	-1
40	0
50	-1
60	0
70	1
80	-1
90	-1
130	-1
140	0
150	1

$\mathbf{X}_2^0$	
t	v
0	0
10	-0.25
20	0
30	0.25
40	0
50	0.25
60	0
90	0.25
100	0
110	-0.25
120	0
130	0.25
140	0
150	-0.25

$\Rightarrow$

$\mathbf{X}_1^1$	
t	v
0	0
10	1
20	0
30	-1
40	0
50	-1
60	0
70	1
80	-1
90	-1
100	-1
110	-1
120	-1
130	-1
140	0
150	1

$\mathbf{X}_2^1$	
t	v
0	0
10	-0.25
20	0
30	0.25
40	0
50	0.25
60	0
70	0.08
80	0.16
90	0.25
100	0
110	-0.25
120	0
130	0.25
140	0
150	-0.25

$\Rightarrow$

$\mathbf{V}$	
V_1	V_2
0	0
1	-0.25
0	0
-1	0.25
0	0
-1	0.25
0	0
1	0.08
-1	0.16
-1	0.25
-1	0
-1	-0.25
-1	0
-1	0.25
0	0
1	-0.25

Figure 3: Original Time Series X_1^0, X_2^0; Initialized Time Series X_1^1, X_2^1; Multidimensional Points V

DEFINITION 2 (SVD). *A matrix $\mathbf{V} = [V_1|V_2|\ldots|V_n] \in \mathcal{R}^{m\times n}$ can be decomposed into a product of three matrices:*

$$SVD(\mathbf{V}) = \mathbf{L}\times\mathbf{\Sigma}\times\mathbf{R}^T = \underbrace{\left[L_1\middle|\ldots\middle|L_n\right]}_{\mathbf{L}(m\times n)} \times \underbrace{\left[\begin{array}{c|c|c} \sigma_1 & \ldots & 0 \\ \hline \vdots & \ddots & \vdots \\ \hline 0 & \ldots & \sigma_n \end{array}\right]}_{\mathbf{\Sigma}(n\times n)} \times \underbrace{\left[\begin{array}{c} R_1^T \\ \hline \vdots \\ \hline R_n^T \end{array}\right]}_{\mathbf{R}^T(n\times n)}$$

Where:

1. $\mathbf{\Sigma}$*: is a $n\times n$ square diagonal matrix that contains strictly positive singular values of $\mathbf{V}$. The diagonal entries σ_i of $\mathbf{\Sigma}$ are the square roots of the eigen values of $\mathbf{V}^T\mathbf{V}$ and are ranked in decreasing order such that $\sigma_1 > \sigma_2 > \ldots > \sigma_n$.*
2. $\mathbf{L}$*: is an $m\times n$ orthogonal matrix whose columns are the orthonormal eigen vectors of $\mathbf{V}\mathbf{V}^T$ ($L^TL = I$, where I is the identity matrix). The eigen vectors of L are computed by solving $Det(\sigma\mathbf{I} - \mathbf{V}\mathbf{V}^T) = 0$ where $Det(\mathbf{X})$ is the determinant of matrix $\mathbf{X}$.*
3. $\mathbf{R}$*: is an $n\times n$ orthogonal matrix having as columns orthonormal eigen vectors of $\mathbf{V}^T\mathbf{V}$ ($R^TR = I$). The eigen vectors of $\mathbf{R}$ are computed by solving $Det(\sigma\mathbf{I} - \mathbf{V}^T\mathbf{V}) = 0$.*
4. *A singular value σ_i defines the variance of vector L_i along dimension R_i^T. Each dimension represents an axis of projection: $var(L_i) = \sigma_i$.*

EXAMPLE 2. *Consider time series X_1^1 and X_2^1 from Figure 3. Figure 4 illustrates the SVD of $\mathbf{V}$.*

3.3.2 Dimensionality Reduction

SVD allows to perform a dimensionality reduction from a dimension n to a lower dimension r. The dimensionality reduction is performed by nullifying the $n-r$ smallest singular values from matrix $\mathbf{\Sigma}$, where $0 < \sigma_r < \sigma_n$. Figure 5 illustrates the dimensionality reduction for $r = n-1$, i.e., the smallest singular value of $\mathbf{\Sigma}$ is nullified. We write $SVD_r(\mathbf{V})$ for the result of a low rank SVD of a matrix $\mathbf{V}$. REBOM uses the low rank SVD for improving the initial imputation of the missing values as described in the next section.

$$SVD(\mathbf{V}) = \underbrace{\begin{bmatrix} 0 & 0 \\ 0.31 & -0.22 \\ 0 & 0 \\ -0.31 & 0.22 \\ 0 & 0 \\ -0.31 & 0.22 \\ 0 & 0 \\ -0.30 & -0.11 \\ -0.30 & 0.04 \\ -0.31 & 0.22 \\ -0.30 & -0.27 \\ -0.30 & -0.75 \\ -0.30 & -0.27 \\ -0.31 & 0.22 \\ 0.00 & 0.00 \\ 0.31 & -0.22 \end{bmatrix}}_{\mathbf{L}} \times \underbrace{\begin{bmatrix} 3.35 & 0 \\ 0 & 0.51 \end{bmatrix}}_{\mathbf{\Sigma}} \times \underbrace{\begin{bmatrix} 0.99 & -0.14 \\ 0.14 & 0.99 \end{bmatrix}}_{\mathbf{R}^T}$$

Figure 4: Example of Singular Value Decomposition

$$SVD_r(\mathbf{V}) = \underbrace{\left[L_1\middle|\ldots\middle|L_m\right]}_{\mathbf{L}(m\times n)} \times \underbrace{\left[\begin{array}{c|c|c|c} \sigma_1 & \ldots & 0 & 0 \\ \hline \vdots & \ddots & \vdots & 0 \\ \hline 0 & \ldots & \sigma_r & 0 \\ \hline 0 & 0 & 0 & 0 \end{array}\right]}_{\mathbf{\Sigma_r}(n\times n)} \times \underbrace{\left[\begin{array}{c} R_1^T \\ \hline \vdots \\ \hline R_n^T \end{array}\right]}_{\mathbf{R}^T(n\times n)}$$

Figure 5: Illustration of Dimensionality Reduction

4. REBOM

REBOM combines the characteristics of a time series with missing values with the characteristics of its most correlated time series to recover blocks of missing values in irregular time series.

4.1 Correlation Ranking Matrix

We define the top-k ranking matrix to capture the correlation between different time series. The correlation is defined over all values of the first vector of the matrix with respect to all values of another vector. The Pearson coefficient is used as a correlation metric. Given two vectors $V_i = [v_{i_1}, v_{i_2}, \ldots, v_{i_n}]$ and $V_j = [v_{j_1}, v_{j_2}, \ldots, v_{j_n}]$ of the same length n, the Pearson correlation coefficient ρ of V_i with respect to V_j is defined as follows:

$$\rho(V_i, V_j) = \frac{cov(V_i, V_j)}{\sqrt{var(V_i)var(V_j)}} = \frac{\sum_{p=1}^{n}(v_{i_p} - \bar{v}_i)(v_{j_p} - \bar{v}_j)}{\sqrt{\sum_{p=1}^{n}(v_{i_p} - \bar{v}_i)^2 \sum_{p=1}^{n}(v_{j_p} - \bar{v}_j)^2}}$$

$$with\ \bar{v}_i = \frac{1}{n}\sum_{p=1}^{n} v_{i_p},\quad \bar{v}_j = \frac{1}{n}\sum_{p=1}^{n} v_{j_p}$$

$\rho(V_i, V_j)$ is undefined if all values of V_i or V_j are equal. The vectors of the correlation ranking matrix are ranked in decreasing order of the Pearson coefficient between the first vector and the remaining vectors.

DEFINITION 3 (TOP-K RANKING MATRIX). *Let $\mathbf{V} = [V_1, V_2, \ldots, V_n]$ be a matrix of n vectors.* $\mathbf{V}^{top\text{-}k} =$

$[V'_1, V'_2, \ldots, V'_k]$ *is defined as the top-k ranking matrix of* $\mathbf{V}$ *with respect to a given vector that contains initialized missing values* $V_q^1 \in \mathbf{V}$ *iff:*

- $\mathbf{V}^{top\text{-}k}$ *contains the k vectors that are most correlated to* V_q^1*:* $\forall V'_i \in \mathbf{V}^{top\text{-}k} \forall V_j \in \mathbf{V} \setminus \mathbf{V}^{top\text{-}k} : |\rho(V'_i, V_q^1)| \geq |\rho(V_j, V_q^1)|$
- *The elements of* $\mathbf{V}^{top\text{-}k}$ *are sorted by their correlation coefficient to* V_q^1 *:* $\forall 1 \leq i < k : |\rho(V'_i, V_q^1)| \geq |\rho(V'_{i+1}, V_q^1)|$

For each matrix $\mathbf{V}^{top\text{-}k}$ *we define a corresponding top-k ranking vector* $\rho_{\mathbf{V}^{top\text{-}k}} = [\rho(V_q^1, V_1^{top\text{-}k}), \rho(V_q^1, V_2^{top\text{-}k}), \ldots, \rho(V_q^1, V_k^{top\text{-}k})]$ *for* V_q^1 *with the* l_1*-norm* $||\rho_{\mathbf{V}^{top\text{-}k}}|| = \sum_{i=1}^{k}(|\rho(V_q^1, V_i^{top\text{-}k})|)$.

Example 3. *Consider Figure 6 with* $\mathbf{V} = [V_1, V_2, V_3, V_4]$ *and top-3 ranking* $\mathbf{V}^{top\text{-}3} = [V_4, V_3, V_1]$ *for* V_4.

$$\mathbf{V}=\begin{bmatrix} 4 & 6 & 3 & 2 \\ 5 & 7 & 1 & 3 \\ 6 & 7 & 9 & 8 \\ 7 & 6 & 8 & 7 \end{bmatrix}, \mathbf{V}^{top\text{-}3}=\begin{bmatrix} 2 & 3 & 4 \\ 3 & 1 & 5 \\ 8 & 9 & 6 \\ 7 & 8 & 7 \end{bmatrix}$$

Figure 6: Example of $\mathbf{V}^{top\text{-}3}$

We get $\rho_{\mathbf{V}^{top\text{-}3}} = [\rho(V_4, V_4), \rho(V_4, V_3), \rho(V_4, V_1)] = [1, 0.93, 0.87]$ *and* $||\rho_{\mathbf{V}^{top\text{-}3}}|| = 2.75$.

4.2 Stepwise Correlation Monotonicity

We prove that REBOM is stepwise monotonic, i.e, choosing a bigger correlation value in the same iteration implies a bigger sum of variances. Lemma 1 states that the l_1-norm of a ranking vector $\rho_{\mathbf{V}}$ is proportional to the sum of the variance of vectors obtained by the application of the low rank SVD. In what follows a submatrix $\mathbf{V}_i = [V_{i_1}, V_{i_2}, \ldots, V_{i_k}]$ that contains k different columns of $\mathbf{V}$ is denoted as $\mathbf{V}_i \in \mathbf{V}$.

Lemma 1. *Let* $\mathbf{V}_i = [V_{i_1}, V_{i_2}, \ldots, V_{i_k}]$ *and* $\mathbf{V}_j = [V_{j_1}, V_{j_2}, \ldots, V_{j_k}]$ *be two different* $m \times k$ *matrices and let* $\mathbf{V}$ *be* $m \times n$ *matrix such that* $n \geq k$ *and* $\mathbf{V}_i, \mathbf{V}_j \in \mathbf{V}$. *Let* $\mathbf{W}_i = [W_{i_1}, W_{i_2}, \ldots, W_{i_k}] = SVD_r(\mathbf{V}_i)$ *and* $\mathbf{W}_j = [W_{j_1}, W_{j_2}, \ldots, W_{j_k}] = SVD_r(\mathbf{V}_j)$ *such that* $V_{i_1} = V_{j_1}$. *The* l_1*-norm of* $\rho_{\mathbf{V}_i}$ *and* $\rho_{\mathbf{V}_j}$ *is proportional to the sum of the variances of* $\mathbf{W}_i$ *and* $\mathbf{W}_j$*:*

$$||\rho_{\mathbf{V}_i}|| > ||\rho_{\mathbf{V}_j}|| \Rightarrow \sum_{p=1}^{k} var(W_{j_p}) > \sum_{p=1}^{k} var(W_{i_p})$$

Lemma 1 states that choosing a matrix with a bigger l_1-norm of the ranking vector implies a higher sum of variances over the vectors obtained by the SVD. Therefore, more correlated vectors of the input matrix yields a higher sum of the variances after the application of $SVD_r()$. Thus, by considering the top-k ranking matrix, the result of $SVD_r()$ maximizes the following objective function:

$$\sum_{V_i \in SVD_r(\mathbf{V})} var(V_i)$$

$$\mathbf{V}=\begin{bmatrix} 4 & 6 & 3 & 2 \\ 5 & 7 & 1 & 3 \\ 6 & 7 & 9 & 8 \\ 7 & 6 & 8 & 7 \end{bmatrix}, \mathbf{W}=\begin{bmatrix} 4.2 & 5.6 & 2.3 & 2.8 \\ 4.9 & 7.1 & 1.4 & 2.4 \\ 6.6 & 6.6 & 8.9 & 7.7 \\ 6.2 & 6.4 & 8.1 & 7.1 \end{bmatrix}$$

Figure 7: Example of a matrix and its SVD_r **transformation**

Example 4. *Consider matrix* $\mathbf{V} = V_1, V_2, V_3, V_4$ *from example 3 and the result matrix of the application of* $SVD_r(\mathbf{V})$ *as shown in Figure 7.*

Let's take the example of $\mathbf{V}_1, \mathbf{V}_2 \in \mathbf{V}$ *where* $\mathbf{V}_1 = \{V_4, V_3, V_1\}$ *and* $\mathbf{V}_2 = \{V_4, V_2, V_1\}$*, and let* $\mathbf{W}_1 = SVD_r(\mathbf{V}_1)$ *and* $\mathbf{W}_2 = SVD_r(\mathbf{V}_2)$.

If we apply the computation with respect to vector V_4*, we get* $||\mathbf{V}_1|| = 2.75, ||\mathbf{V}_2|| = 2.07, \sum_{p=3}^{k} var(W_{1_p}) = 24$ *and* $\sum_{p=1}^{3} var(W_{2_p}) = 9.4$.

Lemma 1 holds for any other matrices $\mathbf{V}_i, \mathbf{V}_j \in \mathbf{V}$.

4.3 Iterative Recovery of REBOM

This section proves that REBOM terminates. In each step we compute the partial correlation ranking for the time series based on the missing values. If this partial ranking is the same as the global ranking, the recovery stops. For all missing values $t \in T_i^0$ (cf. Definition 1) the partial correlation $\widetilde{\rho}(V_i, V_j)$ is defined as follows:

$$\widetilde{\rho}(V_i, V_j) = \frac{\sum_{t=1}^{|T_i^0|}(v_{i_t} - \bar{v}_i)(v_{j_t} - \bar{v}_j)}{\sqrt{\sum_{t=1}^{|T_i^0|}(v_{i_t} - \bar{v}_i)^2 \sum_{t=1}^{|T_i^0|}(v_{j_t} - \bar{v}_j)^2}}$$

Where $|T_i^0|$ is the length of T_i^0. $\widetilde{\rho}(V_i, V_j)$ is undefined if all missing values of V_i or V_j are equal. The partial ranking matrix contains the partially most correlated vectors to the vector that contains the missing blocks to recover.

Definition 4 (Partial ranking matrix). *Given a matrix* $\mathbf{V} = [V_1, V_2, \ldots, V_n]$ *of* n *vectors,* $\widetilde{\mathbf{V}}^{top\text{-}k} = [V'_1, V'_2, \ldots, V'_k]$ *is defined as the top-k partial ranking matrix of* $\mathbf{V}$ *with respect to a given vector* $V_q^1 \in \mathbf{V}$ *iff:*

- $\widetilde{\mathbf{V}}^{top\text{-}k}$ *contains the k vectors that are partially most correlated to* V_q^1*:* $\forall V'_i \in \widetilde{\mathbf{V}}^{top\text{-}k} \forall V_j \in \mathbf{V} \setminus \widetilde{\mathbf{V}}^{top\text{-}k} : \widetilde{\rho}(V'_i, V_q^1) \geq \widetilde{\rho}(V_j, V_q^1)$
- *The elements of* $\widetilde{\mathbf{V}}^{top\text{-}k}$ *are sorted by their partial correlation coefficient to* V_q^1 *:* $\forall 1 \leq i < k : \widetilde{\rho}(V'_i, V_q^1) \geq \widetilde{\rho}(V'_{i+1}, V_q^1)$

The top-k ranking and the top-k partial ranking are used to terminate the iterative recovery process.

Lemma 2 (Termination Condition). *Let* $\mathbf{W}_i^{top\text{-}k} = [W_{i_1}, W_{i_2}, \ldots, W_{i_k}]$ *and let Ranking() be the ranking of vectors inside a matrix. If* $\mathbf{W}_i^{top\text{-}k}$ *and its partial correlation matrix have the same ranking then the algorithm can not anymore create a matrix* $\mathbf{W}_{i+1}$ *with bigger sum of variances along its vectors. Formally:*

$$Ranking(\mathbf{W}_i^{top\text{-}k}) = Ranking(\widetilde{\mathbf{W}}_i^{top\text{-}k}) \Rightarrow$$

$$\sum_{W_{i_j} \in \mathbf{W_i}} var(W_{i_j}) > \sum_{W_{(i+1)_j} \in \mathbf{W}_{i+1}} var(W_{(i+1)_j})$$

After each iteration, REBOM compares the ranking of vectors in the top-k ranking with the ranking of vectors in the top-k partial ranking. If the two rankings are equal, the recovery process terminates. As long as the two rankings are different or one of the two rankings is undefined, the most correlated time series can be used to further improve the accuracy of the recovery.

EXAMPLE 5. *Let* $\mathbf{V}_1 = [V_{1_1}, V_{1_2}, V_{1_3}, V_{1_4}, V_{1_5}]$ *and* $k = 3$. *After each iteration we create matrix* $\mathbf{W}_i$ *with recovered values and compare* $Ranking(\mathbf{W}^{top\text{-}3})$ *with* $Ranking(\widetilde{\mathbf{W}}^{top\text{-}3})$. *Initially,* $\widetilde{\rho}(V_1, V_i)$ *and* $Ranking(\widetilde{\mathbf{V}}^{top\text{-}3})$ *are undefined and thus, REBOM iterates. REBOM terminates after two steps since* $Ranking(\mathbf{W}_2^{top\text{-}3}) = Ranking(\widetilde{\mathbf{W}}_2^{top\text{-}3}) = \{V_1, V_2, V_3\}$. *The vectors of the top-k ranking and top-k partial ranking are highlighted in gray and the recovered values are displayed in bold.*

	$\mathbf{V_1}$				
	V_{1_1}	V_{1_2}	V_{1_3}	V_{1_4}	V_{1_5}
	-1	0.5	0.25	0.75	1
	0	0	0.2	0	0
	-1	0.5	0.25	0	1
	0	0.5	0	0.75	0
	1	0	0	0	0
	0	0	-0.25	0	0
	-1	0.5	0.25	0.75	-1
	-1	0.2	0	0	0.7
	-1	0.4	-0.25	0	0.4
	-1	0.2	0	0.75	0.8
	-1	0.5	0.25	0.75	1
	0	0	0	0	0
	-1	0.5	0.25	0.75	1
$\rho(V_{1_1}, V_{1_i})$	1	-0.69	-0.33	-0.43	-0.46
$\widetilde{\rho}(V_{1_1}, V_{1_i})$	-	-	-	-	-

	$\mathbf{W_1}$				
	W_{1_1}	W_{1_2}	W_{1_3}	W_{1_4}	W_{1_5}
	-1	0.5	0.25	0.75	1
	0	0	0.2	0	0
	-1	0.5	0.25	0	1
	0	0.5	0	0.75	0
	1	0	0	0	0
	0	0	-0.25	0	0
	-1	0.5	0.25	0.75	-1
	-0.5	0.2	0	0	0.7
	-0.8	0.4	-0.25	0	0.4
	-0.5	0.2	0	0.75	0.8
	-1	0.5	0.25	0.75	1
	0	0	0	0	0
	-1	0.5	0.25	0.75	1
$\rho(W_{1_1}, W_{1_i})$	1	-0.78	-0.45	-0.47	-0.41
$\widetilde{\rho}(W_{1_1}, W_{1_i})$	1	-1	1	0.5	0.97

	$\mathbf{W_2}$				
	W_{2_1}	W_{2_2}	W_{2_3}	W_{2_4}	W_{2_5}
	-1	0.5	0.25	0.75	1
	0	0	0.2	0	0
	-1	0.5	0.25	0	1
	0	0.5	0	0.75	0
	1	0	0	0	0
	0	0	-0.25	0	0
	-1	0.5	0.25	0.75	-1
	-0.2	0.2	0	0	0.7
	-0.8	0.4	-0.25	0	0.4
	-0.2	0.2	0	0.75	0.8
	-1	0.5	0.25	0.75	1
	0	0	0	0	0
	-1	0.5	0.25	0.75	1
$\rho(W_{2_1}, W_{2_i})$	1	-0.8	-0.48	-0.46	-0.36
$\widetilde{\rho}(W_{2_1}, W_{2_i})$	1	-1	1	0.5	0.97

Figure 8: Iterative Recovery of REBOM

4.4 Algorithm

Algorithm 1 implements the block recovery of REBOM. First, using the method described in subsection 3.2, $\mathbf{X}^1$ is created by initializing the missing values of $\mathbf{X}^0$. Then, the vectors representing each time series of $\mathbf{X}^1$ are inserted as columns in the matrix of vectors $\mathbf{W}_1$. The vector to recover is inserted as the first column of $\mathbf{W}_1$. The order of the selected vector to recover has no impact on the result of the recovery since only the original vectors are used in the recovery process. Therefore, the proposed recovery is deterministic and does not depend on the order of time series to recover. Next, if the ranking of the top-k ranking matrix is different from the ranking of the top-k partial matrix or one of the rankings is undefined (NAN), the recovery is performed. If $Ranking(\mathbf{W}_j^{top\text{-}k})$ is equal to $Ranking(\widetilde{\mathbf{W}}_j^{top\text{-}k})$ the recovered time series is inserted into the set of recovered time series, i.e, $\widetilde{\mathbf{X}}^j$. Once all time series have been recovered, $\widetilde{\mathbf{X}}^j$ will be returned as the result of REBOM's block recovery.

Input: A set of n time series
$\mathbf{X}^0 = \{X_1^0, X_2^0, \ldots, X_n^0\}$
Output: A set of recovered time series
$\widetilde{\mathbf{X}} = \{\widetilde{X}_1^{j_1}, \widetilde{X}_2^{j_2}, \ldots, \widetilde{X}_n^{j_n}\}$

```
1  begin
2    X^1 = Init(X^0);
3    for each X_i^1 ∈ X^1 do
4      V_i^1 = Extract_val(X_i^1);
5      j = 1;
6      W_j = [V_i^1];
7      for each X̃_p^1 ∈ X̃^1 \ X̃_i^1 do
8        V_p^1 = Extract_val(X̃_p^1);
9        W_j = [W_j, V_p^1];
10     while
       Ranking(W_j^{top-k}) <> Ranking(W̃_j^{top-k}) or
       Ranking(W_j^{top-k})=NAN or
       Ranking(W̃_j^{top-k}) = NAN do
11       LΣR^T = SVD(W_j^{top-k});
12       Σ_r = Reduce_Dim(Σ, n, r);
13       M = L × Σ_r × R^T;
14       W_j = UMV(W_j^{top-k}, M);
15       j+ = 1;
16     X̃_i^j = Add_ts(W_{j_i});
17     X̃^j = {X̃^j} ∪ {X̃_i^j};
18     i+ = 1;
19   return X̃^j;
20 end
```

Algorithm 1: REBOM's Block Recovery

Extract_val() and *Add_ts*() are used respectively to extract values from a time series and to add time stamps to a vector.

The UMV algorithm (cf. Algorithm 2) updates missing values. It accesses the database and uses procedural SQL to determine the indexes of missing values (load_mv_indexes()). The code of this function is described in the the first section of the appendix.

```
1  Algorithm:UMV(V1, V2)
2  begin
3      for each V_i ∈ V1 do
4          T_i^0 = load_mv_indexes(i);
5          for each v_{i_j} ∈ V_i do
6              if position(v_{i_j}) ∈ T_i^0 then
7                  Insert_element(V3, v'_{ij});
                   // Insert v'_{ij} ∈ V2 in row i and
                      column j of V3
8              else
9                  Insert_element(V3, v_{i_j});
10     return V3;
11 end
```

Algorithm 2: Updating Initialized Missing Values

5. EXPERIMENTS

5.1 Experimental Setup

For the evaluation we use real world datasets and synthetic data sets that describe hydrological phenomena of up to 15 million observations produced by sensors in 242 mountain stations. Our hydrological database contains 79 temperature time series, 69 precipitation time series, 48 water level time series, 15 humidity time series, 4 wind speed time series and 3 air pressure time series. The data was provided by an environmental engineering company [23].

We ran experiments to compare the recovery accuracy of REBOM against state-of-the-art techniques.

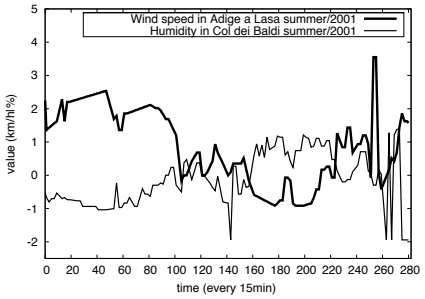
5.2 Experiments with Hydrological Time Series

5.2.1 Restoration of Peaks and Valleys

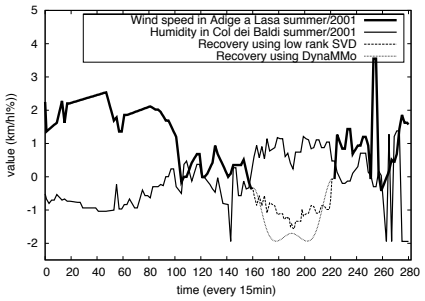
In the first set of experiments, we compare the accuracy of REBOM for the restoration of missing blocks against a non parametric recovery technique that is the (non-iterated) low rank SVD and a parametric recovery technique that is DynaMMo [10]. These two techniques are the most accurate techniques for the recovery of blocks of missing values in time series. We ran our experiment on wind speed and humidity time series. Figure 9(a) shows two time series measured during summer season (one measurement every 15 minutes) in two different areas of the region of Alto-Adige (Italy). We drop a block of values for $t \in]160, 220[$ and restore it using the low rank SVD and DynaMMo. The dropped block includes a valley with a small peak.

The recovery of the two techniques is shown in Figure 9(b). The low rank SVD is only able to detect part of the trend of the missing block, i.e., only a valley is recovered. The shape of the recovered valley resembles the shape of the block that belongs to the same time interval of the missing block in the other time series. DynaMMo is able to detect the entire trend of the missing block, i.e., a valley containing a small peak. However the shape of the original block is not accurately restored. The recovered block looks similar to a smooth spline that contains a small peak. Since we use only tow time series REBOM will not iterate. Therefore, the recovery of REBOM is similar to the recovery of the low rank SVD.

We add a second humidity time series to the experiment to compare the block recovery of REBOM against DynaMMo (see Figure 10). The result of Figure 10(b) shows that the recovery of DynaMMo does not change by the addition of a third time series because DynaMMo cannot use more than one reference time series in the recovery process. REBOM exploits the two humidity time series in the recovery process. It uses the history of the wind speed time series together with the correlation with respect to the two humidity time series to recover the missing block. Both the trend and the shape of the missing block are accurately recovered. Adding more correlated time series will further improve the block recovery of REBOM (see Figure 12).



(a) Time Series Measured in Two Different Areas



(b) Recovery of a Removed Block

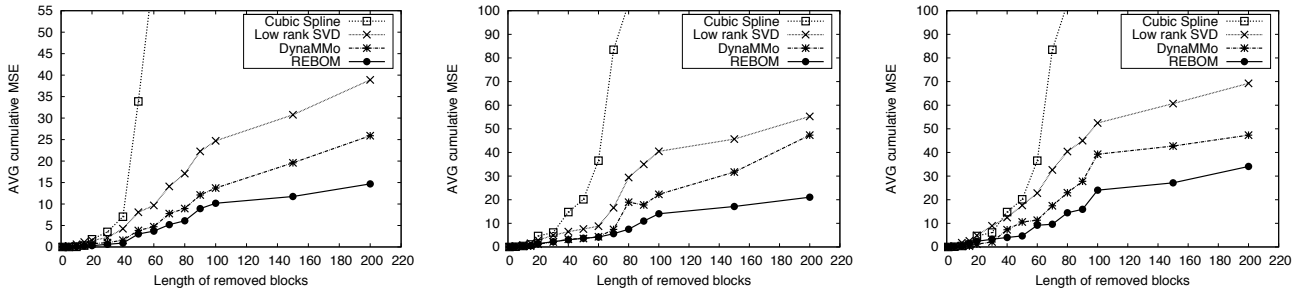
Figure 9: Recovery using Low Rank SVD and DynaMMo

We run a second set of experiments in which we compare the block recovery error using the Mean Square Error (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(w_i - {v_i}^+)^2$$

where w is the recovered value, v^+ is the original value and n is the number of deleted observations.

Figure 11 shows the cumulative recovery error for removed blocks of values of increasing length: we set a starting timestamp, we vary the length of the removed block and we compute the cumulative MSE of each block. The x-axis represents the length (number of values) of the removed block to recover and the y-axis represents the average cumulative MSE. The experiments in Figures 11(a) and 11(b) are executed respectively on six different temperature time series with 1000 values each measured in region of Alto Adige and four different humidity time series with 1000 values each measured in the region of Vipetino. For these two experiments, we remove a block from one time series only while the other time series are complete. The results in both exper-

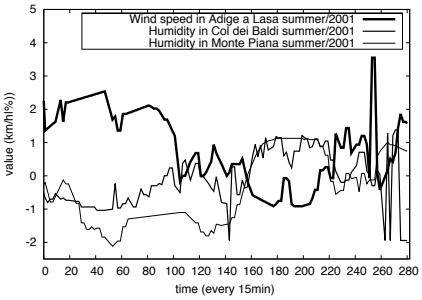


(a) Average Cumulative Error for Successive Removed Blocks in One Temperature Time Series. The correlated temperature time series used in the recovery are complete.

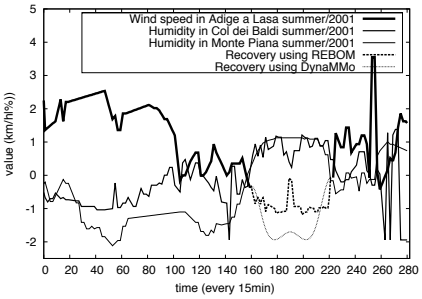
(b) Average Cumulative Error for Successive Removed Blocks in One Humidity Time Series. The correlated humidity time series used in the recovery are complete.

(c) Average Cumulative Error for Successive Removed Blocks in One Humidity Time Series. The correlated humidity time series used in the recovery contain missing values.

Figure 11: Recovery of Blocks of Different Lengths



(a) Time Series Measured in Three Different Areas



(b) Recovery of a Removed Block

Figure 10: Recovery Using REBOM and DynaMMo

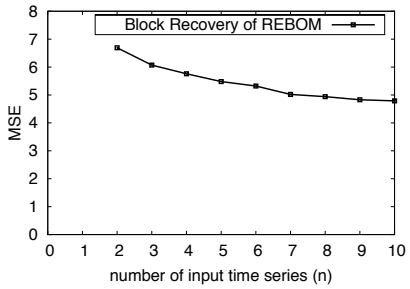
iments show that REBOM outperforms the low rank SVD and DynaMMo for the recovery of successive blocks of missing values and cubic spline is off the scale. For blocks of up to 100 removed values, the recovery error of REBOM slightly increases with the number of removed values. For blocks of more than 100 removed values, the error becomes almost stable and is not anymore affected by the number of removed values. In contrast, the recovery error of DynaMMo and the low rank SVD increases with the length of removed blocks. The small cumulative recovery error of REBOM is due to the use of different correlated time series at every iteration of the algorithm. The experiment of Figure 11(c) is executed on four humidity time series of 1000 values each. The first time series is complete, the second time series contains a missing block in the time range $[0, 100]$, the third time series contains a missing block in the time range $[100, 200]$ and the fourth time series contains a missing block in the time range $[200, 300]$. We execute the same process performed in the experiment of Figure 11(b) for the complete correlated humidity time series. The experiment shows that, compared to the result of Figure 11(b), the recovery accuracy of REBOM, DynaMMo and low rank SVD gets worse when using multiple time series with missing values. The recovery accuracy of REBOM is still better than the one of the other techniques.

In the experiment of Figure 12, we use different correlations and number of input time series (n) to evaluate the impact on the recovery MSE. We vary n and we compute the MSE of REBOM for the same block containing 90 missing values. Figure 12(a) shows that in the case of time series of high correlation ($1 \geq |\rho| > 0.7$), the MSE of REBOM decreases only slightly as n grows. REBOM is able to restore the missing block using a small number of highly correlated input time series. This result is explained by the fact that, for highly correlated time series, the starting top-k ranking matrix is similar to the partial ranking matrix. Therefore, the recovery of REBOM converges quickly. Figure 12(b) shows that, using more time series of moderate correlation ($0.7 \geq |\rho| > 0.4$), the MSE of REBOM decreases linearly. REBOM uses all the time series to perform the most accurate recovery. Figure 12(c) illustrates that, the MSE increases for input time series with low correlated time series ($0.4 \geq |\rho| > 0$).

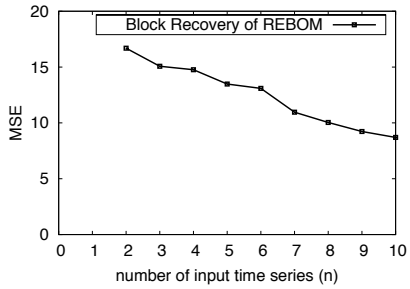
In the experiment of Figure 13 we set n to 10 and we vary the number of time series in the top-k ranking matrix. In Figure 13(a) the minimum MSE is reached for $k \in [2, 4]$. In Figures 13(b) and 13(c), the minimum recovery MSE is reached for a single value that is respectively $k = 4$ and $k = 2$. Again, the recovery accuracy of REBOM decreases for time series with low correlation, i.e., $0.4 \geq |\rho| > 0$, in the top-k ranking matrix.

5.2.2 *Invariance to Initialization Method*

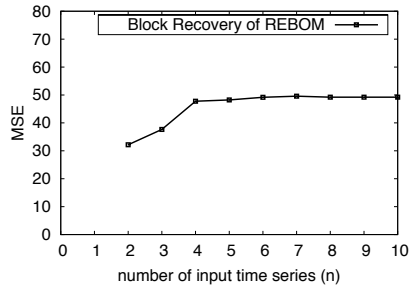
We run this experiment to test the impact of the initialization method on the block recovery of REBOM. Figure 14 shows that with different initialization techniques, REBOM needs more iterations to reach the minimum recovery error. Compared to our initialization method, a linear spline initialization needs twice the number of iterations to reach the minimum recovery error. Using a k Nearest Neighbor initialization, REBOM needs 2.5 times more iterations than



(a) Impact of Input Time Series in the Recovery MSE. Each time series has a correlation value: $1 \geq |\rho| > 0.7$

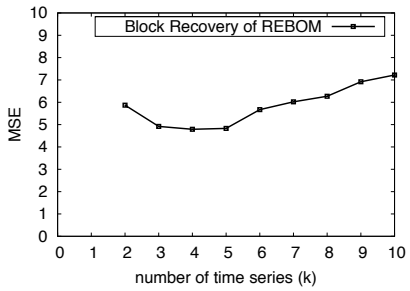


(b) Impact of Input Time Series in the Recovery MSE. Each time series has a correlation value: $0.7 \geq |\rho| > 0.4$

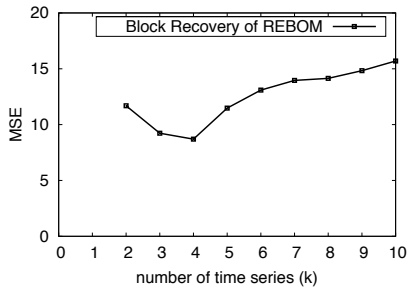


(c) Impact of Input Time Series in the Recovery MSE. Each time series has a correlation value: $0.4 \geq |\rho| > 0$

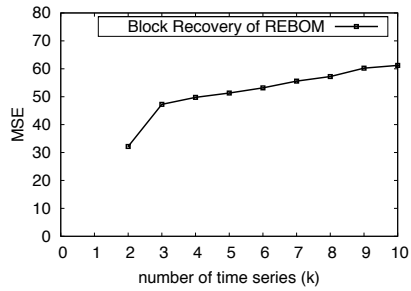
Figure 12: Impact of n in the Recovery MSE of REBOM



(a) Impact of k in the Recovery MSE. Each time series has a correlation value: $1 \geq |\rho| > 0.7$



(b) Impact of k in the Recovery MSE. Each time series has a correlation value: $0.7 \geq |\rho| > 0.4$



(c) Impact of k in the Recovery MSE. Each time series has a correlation value: $0.4 \geq |\rho| > 0$

Figure 13: Recovery MSE using different number of time series in the top-k ranking matrix

our initialization technique to reach the same recovery error. Thus, the accuracy of REBOM is independent from the initialization method. However, our initialization initialization method provides a faster recovery of blocks of missing values.

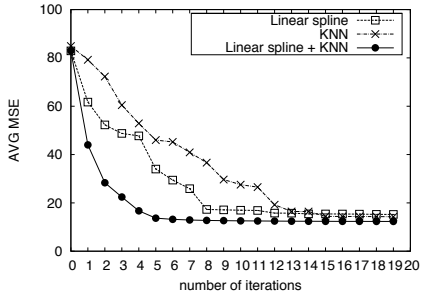


Figure 14: Number of Iterations using Different Initialization Techniques

5.2.3 *Running Time Performance*

The REBOM implementation uses the Golub/Kahan decomposition algorithm [24] and has a run time complexity of $\mathcal{O}(\#iterations \times (4n^2k + 8nk^2 + 9k^3))$, where n is the length of the longest time series and k is the number of vectors of $\mathbf{V}^{top\text{-}k}$. The complexity of building $\mathbf{V}^{top\text{-}k}$ is the cost of computing k times ρ between two time series and that is $\mathcal{O}(kn^2)$. Therefore, the total complexity of using REBOM is $\mathcal{O}(\#iterations \times (5n^2k + 8nk^2 + 9k^3))$. Figure 15 compares the total running time of REBOM against DynaMMo that has a complexity of $\mathcal{O}(\#iterations \times (kn^3))$. 3000 different time series were created by extracting 1000 observations from 15 different temperature time series.

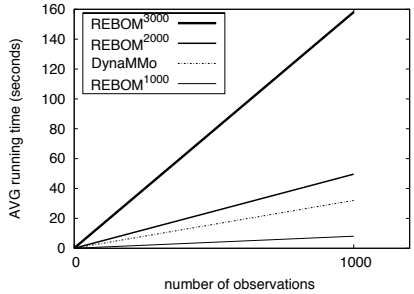
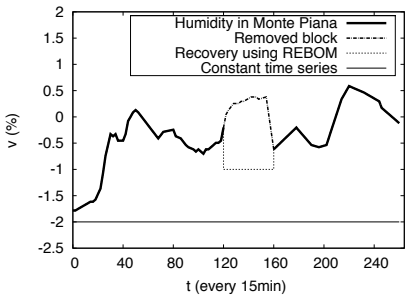


Figure 15: Average Running Time Comparison

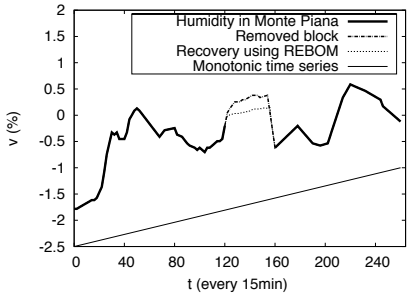
Figure 15 shows the average running time comparison performed on the created time series for the recovery of blocks containing 200 missing values. We set the value of k to four, since we reached the optimal recovery accuracy with this value. The result of this experiment shows that with 1500 time series, REBOM is faster than DynaMMo. With a higher number of input time series, the performance of REBOM starts to be slower than DynaMMo.

5.2.4 Recovery Using Linear Time Series

In the experiment of Figure 16, we show the impact of using extremely irregular time series. We take as input a humidity time series measured in spring 2001 from which we remove a block for $t \in]120, 160[$, a constant time series, and a monotonic time series. The result of Figure 16(a) shows that, since the correlation between the humidity time series and the constant time series is undefined (all values are equal), REBOM performs a bad recovery. In Figure 16(b), the humidity time series and the monotonic time series are correlated. Therefore, both time series are used to recover the type of the missing block. The recovered block has an increasing monotonic shape that looks similar to the monotonic time series. In the experiment of Figure 16(b), both the type and the shape of the missing block are accurately recovered. The application of DynaMMo in the experiment would set all the recovered values to 0.



(a) Recovery of REBOM using lines of function v=c, where c is a constant. The result of the recovery is the same for any given value of c



(b) Recovery of REBOM using lines of function v=at+b, where a =0.5 and b=-2.5

Figure 16: Impact of Extremely Irregular Time Series in the Recovery of REBOM

5.3 Experiments with Synthetic Regular Time Series

This subsection describes a set of experiments conducted with synthetic data. We compare the block recovery of REBOM against DynaMMo.

5.3.1 Different Amplitudes

Figure 17 compares the recovery of the two techniques for two regular time series having different amplitudes. The first time series is a sin(t) wave and the second time series is a sine wave multiplied by a negative scaling factor, i.e., -0.25*sin(t). For $t \in]70, 110[$, we drop a block from sin(t) and we recover it using REBOM and DynaMMo. Both techniques are able to accurately recover the missing block. REBOM uses the correlation between the two time series in order to determine the shape of the missing block, i.e, a peak. The amplitude of the missing peak is determined using the amplitude of the existing peaks from sin(t). The two techniques perform an accurate recovery for any other scaling factor of the second wave.

5.3.2 Shifted Peaks

Figure 18 shows two regular time series shifted in time, i.e., sin(t) and cos(t). For $t \in]70, 110[$, we drop a block from sin(t) and we recover it using REBOM and DynaMMo. REBOM is applied without initial alignment of the two time series. As expected, DynaMMo outperforms REBOM in recovering the missing block. DynaMMo is able to compute the periodicity model and performs a good block recovery. However, REBOM recovers a block that is only influenced by the shape of the block in cos(t) for $t \in]70, 110[$, i.e., a peak followed by a valley. For shifted time series, REBOM is not able to use the history of sin(t) in the recovery process. The decomposition performed by our technique is sensitive to the row position of values inside the $\mathbf{V}^{top\text{-}k}$ matrix. In order to overcome this problem, an initial alignment between the two time series must be performed in a preprocessing step (cf. Subsection 3.2).

6. CONCLUSION

This paper studies the recovery of blocks of missing values in irregular time series. We develop an iterative greedy algorithm called REBOM, that uses at every iteration the most correlated time series to the time series that contains the missing blocks to reconstruct missing peaks and valleys. Empirical studies on real hydrological data sets demonstrate that our algorithm has the most accurate block recovery among existing techniques. In future work, it is of interest to examine the impact of using the recovered time series in the recovery process instead of the original ones. It is also foreseen to investigate the impact the global correlation on the recovery accuracy together with the local correlation. Another promising direction, is to progress the interaction with the database and develop an SQL based recovery solution that reduces the number of I/O's.

Acknowledgments

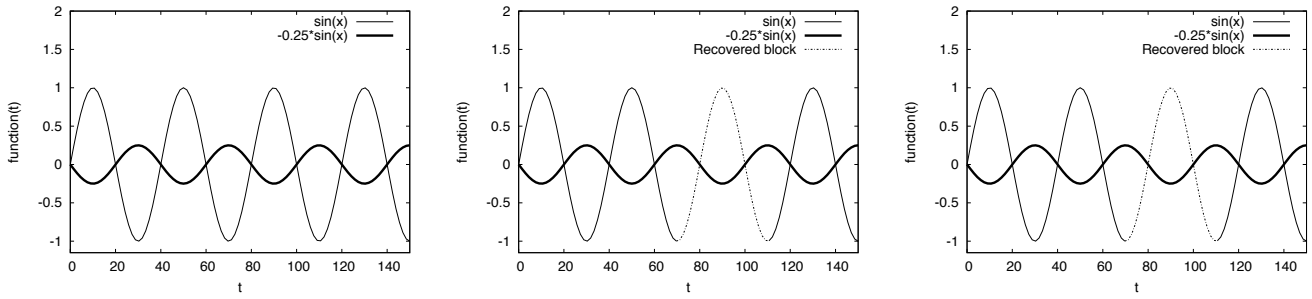
The authors would like to thank HydroloGis company [23] for providing the hydrological datasets that we have used for our experiments, and Michal Koltonik for his contribution in the implementation of REBOM. We wish also to give special thanks to the anonymous reviewers for their insightful comments.

7. REFERENCES

[1] Mueen, A., Nath. S., and Liu, J., : *Fast Approximate Correlation for Massive Time-series Data*, in SIGMOD, 2010

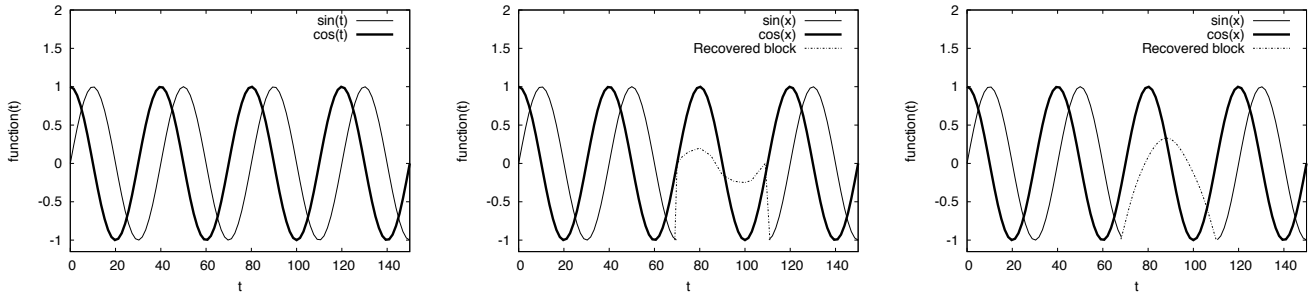
[2] Srebro, N., and Jaakkola, T., : *Weighted Low-Rank Approximations*, in ICML, 2003

[3] Tsechansky, M.S., and Provost, F., : *Handling Missing Values when Applying Classification Models*, in JMLR, 2007



(a) Original Waves (b) Recovery of REBOM (c) Recovery of DynaMMO

Figure 17: Recovery of DynaMMO and REBOM for Time Series of Different Amplitudes



(a) Original Waves (b) Recovery of REBOM (c) Recovery of DynaMMO

Figure 18: Recovery of DynaMMO and REBOM for Shifted Time Series

[4] Romero, V., and Salmerón, F., : *Multivariate Imputation of Qualitative Missing Data using Bayesian Networks*, in SMPS, 2004

[5] Harvey, M., Carman, M.J., Ruthven, I., and Crestani, F., : *Bayesian Latent Variable Models for Collaborative Item Rating Prediction*, in CIKM, 2011

[6] Srebro, N., and Jaakola, T., : *Weighted Low-Rank Approximations*, in ICML, 2003

[7] Li, L., MacCann, J., Pollard, N., and Faloutsos, C., : *DynaMMo: Mining and Summarization of Coevolving Sequences with Missing Values*, in KDD, 2009

[8] Jain, A., Chang, E.Y., and Wang, Y.F., : *Adaptive Stream Resource Management using Kalman Filters*, in SIGMOD, 2004

[9] Ding, H., et al. : *Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures*, in PVLDB, 2008

[10] Chen, Q., Chen, L., Lian, X., and Yu, J.X., : *Indexable PLA for Efficient Similarity Search*, in VLDB, 2007

[11] Gelaman, A., : *Data Analysis using Regression and Multilevel/Hierarchical Models*, Publisher: Cambridge University Press, 1 edition, 2006

[12] Yi., B.K., Sidiropoulos, N.D., Johnson, T. , Jagadish, H.V., Faloutsos, C., and Biliris, A., : *Online Data Mining for Co-Evolving Time Sequences*, in ICDE, 2000

[13] Troyanskaya, O., et al : *Optimal Multi-step k-Nearest Neighbor Search*, in J. Bioinformatics, 2001

[14] Seidl, T., and Kriegel, H.P., : *Optimal Multi-step k-Nearest Neighbor Search*, in SIGMOD, 1998

[15] Kurucz, M., Benczur, A.A., and Csalogany, K., : *Methods for Large Scale SVD with Missing Values*, in KDD, 2007

[16] Meyer, C.D., : *Matrix Analysis and Applied Linear Algebra*, Publisher: SIAM-Society for Industrial and Applied Mathematics, pages 412-417 and 489-504, 2000

[17] Kalman, D., : *A Singularly Valuable Decomposition: The SVD of a Matrix*, in J. College Mathematics, 1996

[18] Brand, M., : *Incremental Singular Value Decomposition of Uncertain Data with Missing Values*, in ECCV, 2002

[19] Alter, O., and al., : *Singular value decomposition for genome-wide expression data processing and modeling*, in PNAS, 2000

[20] Mohan, M., Chen., Z., and Weinberger, K., : *Web-Search Ranking with Initialized Gradient Boosted Regression Trees*, in JMLR, 2011

[21] He, Y., : *Missing Data Imputation for Tree-Based Models*, PhD dissertation, 2006

[22] Ding, Y., and Simonoff., J.S., : *An Investigation of Missing Data Methods for Classification Trees Applied to Binary Response Data*, in JMLR, 2010

[23] HydroloGIS company, available for online access at: `http://www.hydrologis.eu/`

[24] Erricos, J., : *Handbook on Parallel Computing and Statistics*, Book, Chapter 4, 2005

[25] Lagarias, J.C., : *Monotonicity Properties of the Toda Flow, the QR-Flow, and Subspace Iteration*, in SIAM J. Matrix Analysis and Applications, 1991

APPENDIX

A. FUNCTION COMPUTING MISSING TIME STAMPS

We consider two relations:

- *Observation* (series_id, ts, val) that stores the values of observations, where series_id is the id of time series, ts and val are respectively the time stamp and value of observations
- *Series* (id, granul) that stores information about time series, where id is the id of time series and granul is the granularity of time series, i.e., a time series has a granularity of two if the observations occur every two minutes.

Given these two relations, we define function *load_mv_indexes()* that efficiently finds the indexes of all missing time stamps. This function uses the granularity of each time series in order to create a sequence of incremental granularities, e.g., {2,4,6,...}. Then, the set difference between the sequence of granularities and the existing time stamps gives the indexes of missing time stamps. *load_mv_indexes()* is executed as an SQL function on the database server side.

```
FUNCTION load_mv_indexes (in_series_id IN
                          INTEGER) AS
    ts_lst  INTEGER;
    gran    INTEGER;
    out_mv_indexes   INTEGER;

  BEGIN
    SELECT granul
    INTO gran
    FROM Series
    WHERE id=in_series_id;

    SELECT MAX(ts)
    INTO ts_lst
    FROM Observation
    WHERE series_id=in_series_id;

    SELECT ts
    BULK COLLECT INTO out_mv_indexes
    FROM (
      SELECT * FROM (
        SELECT (level-1)*gran ts
        FROM dual
        CONNECT BY LEVEL <= (ts_lst+gran)/gran
      MINUS
        SELECT ts
        FROM Observation
        WHERE series_id=in_series_id
      ) ORDER BY 1
    );
  RETURN(out_mv_indexes);
  END;
```

B. PROOF SKETCHES

B.1 Lemma 1

PROOF. We prove that our algorithm is stepwise monotonic. We perform this proof by showing that the correlation matrix used is monotonic at every step of the algorithm. i) From Def. 2 (SVD) we know that the singular values define the variances along the vectors. ii) From the definition of the top-k ranking matrix we know that at every step of SVD, we take the matrix with the biggest l-1 norm of correlation. iii) From the definition of $SVD_r()$ we know that only the smallest variance will be nullified and the biggest ones will be kept. Using i), ii) and iii) we can deduce that our algorithm takes the biggest $||\rho_{\mathbf{V}_i}||$ in order to compute the biggest $\sum_{V_{i_j} \in \mathbf{W}_i} var(V_{i_j})$ where $\mathbf{W}_i = SVD_r(\mathbf{V}_i)$. Therefore, the bigger the correlation is, the bigger sum of variances we will obtain. This implies that the correlation used by the algorithm is stepwise monotonic. □

B.2 Lemma 2

PROOF. We prove that our algorithm terminates after finding the matrix that has the maximum sum of variances along its vectors. We perform this proof by showing that the iterative refinement of missing values satisfies the following two properties:

- a) *finite number of rankings*: i) From Def. 2 (SVD) we know that the variance values obtained by SVD are ranked in increasing order in matrix $\mathbf{\Sigma}$. ii) From [25] we have that the singular values obtained by SVD are monotonic. Using i) and ii) it follows that the variance obtained by the decomposition is monotonic and thus: $W_{1_j} \in \mathbf{W}_1^{top\text{-}k} \wedge W_{2_j} \notin \mathbf{W}_2^{top\text{-}k} \Rightarrow W_{3_j} \notin \mathbf{W}_3^{top\text{-}k}$ where $\mathbf{W}_2^{top\text{-}k} = SVD_r(\mathbf{W}_1^{top\text{-}k})$ and $\mathbf{W}_3^{top\text{-}k} = SVD_r(\mathbf{W}_2^{top\text{-}k})$. Therefore, the number of rankings generated by our algorithm is finite and this property is satisfied.
- b) *ranking of a matrix determine the result of* $SVD_r()$: Let R_i be the ranking of matrix $\mathbf{W}_i$, $\widetilde{R}_i$ be the partial ranking of matrix $\mathbf{W}_i$ and R_{i+1} be the ranking of matrix $\mathbf{W}_{i+1}$ where $\mathbf{W}_{i+1} = SVD_r(\mathbf{W}_i)$. i) We have from Def. 3 that the correlation value determines the ranking inside a matrix and then $||\rho_{\mathbf{W}_i}|| = ||\rho_{\mathbf{W}_{i+1}}|| \Rightarrow R_i = R_{i+1}$. ii) Since UMV algorithm (cf. Subsection 4.4) is updating only the missing values of the matrix, then: $\widetilde{R}_i$ determines $||\rho_{\mathbf{W}_{i+1}}||$ and it follows that: $R_i = \widetilde{R}_i \Rightarrow ||\rho_{\mathbf{W}_i}|| = ||\rho_{\mathbf{W}_{i+1}}||$. Using i) and ii) we deduce by transitivity that: $R_i = \widetilde{R}_i \Rightarrow R_i = R_{i+1}$ and therefore, this property is satisfied.

Properties a) and b) hold for matrices whose vectors are correlated. It follows the proof for this lemma. □