MULTIVARIATE TIME SERIES IMPUTATION USING GENETIC PROGRAMMING

Damares Crystina Oliveira de Resende
Federal University of Pará - UFPA
Av. Augusto Corrêa, 01 - 66075-110 - Belém - PA, Brazil
damares.resende@itec.ufpa.br

Ádamo Lima de Santana
Federal University of Pará - UFPA
Av. Augusto Corrêa, 01 - 66075-110 - Belém - PA, Brazil
adamo@ufpa.br

Antônio F. Lavareda Jacob Júnior
State University of Maranhão - UEMA
Cidade Universitria Paulo VI - 65800-000 - São Luís - MA, Brazil
jacobjr@engcomp.uema.br

Fábio Manoel França Lobato
Federal University of Western Pará - Ufopa
Rua Vera Paz - 68035-110 - Santarém - PA, Brazil
fabio.lobato@ufopa.edu.br

ABSTRACT

Time series have been used in several applications such as process control, environment monitoring, financial analysis and scientific researches. However, in the presence of missing data, this study may become more complex due to a strong break of correlation among samples. Therefore, this work proposes an imputation method for time series data, using a Genetic Programing algorithm based on a multi-criteria fitness function. The heuristics studied build an interpretable regression model that explores time series statistical features such as mean, variance and autocorrelation, and makes use of the interrelation among multivariate time series to estimate missing values. As a consequence, to better understand the missing data pattern, analysis can later use the model built to extract knowledge. Results show that the studied approach promising and is capable of imputing data without losing the dataset's statistical properties.

KEYWORDS. Time series analysis. Data imputation. Genetic programming. Missing Data.

Paper topics: MH. OA.

1. Introduction

Time series have been largely used in many areas of application such as economics, business, social sciences and natural sciences. This data is built from a sequence of random observations taken sequentially over time, which can be described as a stochastic process where adjacent examples have strong correlation and little variation [Box et al. 2008]. The particularities of a time series make it a special case of study since characteristics such as seasonality and smooth variations may be a problem for traditional data analysis methods [Shumway and Stoffer 2010]. In addition to that, the presence of missing values in this type of data makes its inquiry even more complex due to a strong break of correlation among samples.

Given this issue, this work proposes a Genetic Programming algorithm to estimate missing values in multivariate time series data, called GPImpute. An evolutionary methodology was chosen since it is highly flexible and relatively fast in terms of exploration of the solution space [Freitas 2013]. Moreover, a Genetic Programming algorithm is able to build regression functions from data examples, what makes it a good approach for time series forecasting and control applications.

The GPImpute is based on a multi-criteria fitness function that considers three important metrics, which characterize a time series distribution: mean, variance and auto-correlation. This approach was chosen aiming to preserve the original properties of the data distribution after imputation, and it is based on [Figueroa Garcia et al. 2010] methodology for treating missing values in univariate time series. Furthermore, the proposed algorithm builds a comprehensive regression model that can be used by data analysts to understand the missing pattern in the data distribution.

This article is divided as follows: Section 2 reviews some work related to our proposal. Section 3 gives a short overview on time series data analysis. Section 4 describes the methodology used in this work. Section 5 presents some experiments performed. Section 6 reports the results obtained. And finally, Section 7 discusses our conclusions and future research.

2. Related Work

A well accepted approach to deal with missing values in time series is through data imputation. [Junger and de Leon 2015], for instance, proposed a method to treat missing data in multivariate time series by making use of an EM (Expectation Maximization) algorithm, [Liu and Molenaar 2014] did the same, but with models based on autorregressive vectors. [Wellenzohn et al. 2014], on the other hand, proposed a *hot deck* imputation method for streams of time series that are similar to each other. [Cai et al. 2015] treated imputation as an optimization problem using tensor factorization to build a model to regress values, and [Figueroa Garcia et al. 2010]

Although, due to the complexity accounted to the estimation of missing data, many authors chose to invest in evolutionary algorithms to build models for this application. [Lobato et al. 2015b] for example, proposed a Genetic Algorithm to treat missing values in classification datasets, making use of both numeric and categorical data. [Patil and Bichkar 2010], also studied missing data in pattern classification databases. The authors developed a hybrid algorithm that makes use of a Genetic Algorithm and decision tree learning to impute data. [Tran et al. 2015], on the other hand, invested in a Genetic Programming Algorithm to treat missing values, achieving great results in both prediction and classification accuracy. The authors pointed out that this evolutionary algorithm is an excellent choice for regression.

Exploring multi-criteria fitness functions, [Lobato et al. 2015a] proposed a Genetic Algorithm for estimating missing values in mixed-attribute datasets, an approach also used by [Garcia et al. 2011] to treat missing data in multivariate databases and by [Figueroa Garcia et al. 2010] to impute data in univariate time series. The later, together with [Tran et al. 2015], are the main methodologies this work is based on.

3. Time Series Analysis

Time series are a sequence of random observations taken over time. This structure is commonly used in a variety of applications such as economics and business, where stock markets are analyzed; natural sciences, where the climate conditions are observed and forecasted; or engineering, where processes are studied and controlled.

Despite the large use of time series data, this type of dataset is more complex to analyze. It has particular properties inherited from time dependency that may be a problem for traditional data analysis methods to deal with. Since in a time series, adjacent examples have small difference and are strongly correlated, conventional statistical methods that consider that these observations are independent and uniformly distributed may harm prediction's accuracy [Shumway and Stoffer 2010].

In order to treat missing values in a time series data, characteristics such as seasonality, correlation among samples and smooth variations between examples must be considered. Several researches explore these properties [Cai et al. 2015], [Honaker and King 2010], [Junger and de Leon 2015], [Figueroa Garcia et al. 2010]. The later makes use of a Genetic Algorithm to estimate missing values in univariate time series. In this work, the authors combine statistical features of the data in a multi-criteria fitness function for conserving the original aspects of the dataset.

In [Figueroa Garcia et al. 2010], basically three metrics are used in the fitness function: mean, variance and auto-correlation. The present work uses the same heuristic, however the algorithm coded handles multivariate time series data, and the evolutionary approach was replaced by a Genetic Programming algorithm. This methodology was chosen because this method is able to build regression functions from example data, what makes it an excellent choice for regression applications [Tran et al. 2015]. The mean, variance, auto-covariance and auto-correlation functions are described in Equations 1, 2, 3 and 4 respectively.

$$\mu_x = \frac{1}{n}\sum_{t=1}^{n} x_t \tag{1}$$

$$\vartheta_x = \frac{1}{n-1}\sum_{t=1}^{n}(x_t - \mu_x)^2 \tag{2}$$

$$\gamma_(h) = \frac{1}{n}\sum_{t=1}^{n-|h|}(x_{t+|h|} - \mu_x)(x_t - \mu_x), \quad -n < h < n. \tag{3}$$

$$\rho(h) = \frac{\gamma(h)}{\gamma(0)}, \quad -n < h < n. \tag{4}$$

The mean and variance indicated in equations 1 and 2 are the standard calculations for these measures. The mean is calculated by summing all samples (x_t) and then dividing it by the total number of samples (n) in the time series; and the variance is calculated by the sum of the difference of each sample by the mean of the distribution divided by the total number of samples minus one.

The calculation of the auto-covariance and auto-correlation measures are special cases. The auto-correlation is a stochastic process represented by the distance of an observation x_t to another observation x_{t+h}, where h is the lag. This auto-correlation is measured by the ratio between the auto-covariance of x_t and a sample $x_t t + h$ at lag h, and the auto-covariance of x_t with no lag. The auto-correlation value must lie between +1 and -1, where +1 indicates a perfect positive correlation and -1 indicates a perfect negative correlation [Box et al. 2008]. Both auto-covariance and auto-correlation functions require that the time series is complete in order to be computed.

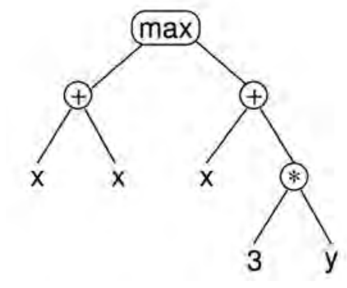
4. Genetic Programming

Genetic programming is an evolutionary computational method used in complex optimization problems and it is typically applied to machine learning tasks such as classification and prediction [Eiben and Smith 2003]. This methodology is able to solve problems automatically based on high level premises without requiring the user to previously know or specify the solution's structure [Poli et al. 2008].

According to [Tran et al. 2015], a genetic programming algorithm is capable of learning definitions of functions based on example data, what makes it an excellent approach for regression tasks. Moreover, this type of algorithm codes the individuals in a non-linear structure, a tree or graph for example, a characteristic that makes it a viable option to solve non-linear problems.

Like other evolutionary algorithms, genetic programming is based on Darwin's premise "The fittest survive", and each individual is a possible solution for the problem. In a tree-based algorithm, each candidate is a program where the internal nodes of the tree are operations and the leaves are variables. The operations in internal nodes, called functions, can be of any type such as arithmetic, trigonometric and boolean; the variables, called terminals, can be input variables such as x and y, constants and 0-arity functions such as go_left and $rand()$. Figure 1 illustrates an example of program in genetic programming algorithms that represents the program $max(x+x, x+3\times y)$.

Figure 1: Example of individual in Genetic Programming.



Source: [Poli et al. 2008].

5. GPImpute - The proposed algorithm

The GPImpute algorithm aims to estimate missing values in a time series database. It is coded on the top of a Genetic Programming algorithm based on a multi-criteria fitness function, where each parameter deals with a particular specificity of a time series in order to maintain its original characteristics after imputation.

An evolutionary method was chosen due to its high flexibility and fast exploration capacity of the search space [Freitas 2013]. Besides, as said before, a Genetic Programming algorithm is able to construct functions from data examples, a feature that makes it an excellent choice for regression [Tran et al. 2015] and then for time series forecasting and control applications. The GPImpute was coded using the ECJ package [Luke et al. 2004], a Java software with implementations of evolutionary algorithms. The parameters used are summarized in Table 1.

The parameters chosen are based on a parametrization process, where the values 64, 128 and 256 for number of generations, and the values 1024, 2048 and 4096 for number of individuals were tested, totalizing 9 tests. Then it was analysed the fitness conversion and the algorithm response in terms of prediction accuracy and computational time. As a result, the best combination found was the value 128 for number of generations and the value 2048 for number of individuals. The function set consists of trigonometric and arithmetic operations. Further, each terminal is a feature not being regressed. Hence, if an attribute a is currently being analyzed for imputation, all other features in the database, except feature a, can be considered as a terminal. Likewise, in order to give more flexibility to the function, random values are also considered as a possible terminal.

Finally, according to [Wolfgang et al. 1998], since the size of the population is normally high, what already guarantees a certain variability, in order to converge, the mutation rate on a genetic programming algorithm must be less than 5%. Moreover, according to [Eiben and Smith 2003], to increase efficiency, the crossover rate must be proportional to the number of individuals

Parameters	**Values**
Function set	$+$, $-$, $\times$, $/$ (protected division), exp, sin, cos, abs
Variable terminals	all attributes except the one being regressed
Constant terminals	random values
Population size	2048
Initialization	ramped half-and-half
Generations	128
Crossover probability	75%
Reproduction probability	20%
Mutation probability	5%
Selection type	tournament (size = 5)

Table 1: Genetic Programming Parameters.

in line with the rule of thumb. For 2048 number of individuals, the selection rate should be around 16%. The mutation and crossover operations are applied to subtrees of an individual. For mutation operation, the subtree of an arbitrary chosen node is replaced by a random generated subtree. For crossover operation, subtrees of arbitrary chosen nodes of the selected individuals are interchanged.

5.1. Multi-criteria fitness function

In order to maintain the original properties of a time series, the fitness function has multiple parameters. Basically, three characteristics were chosen: mean, variance and auto-correlation, each one being responsible for keeping one basic aspect of a time series.

The fitness function is given by Equation 5. The goal of the algorithm is to *minimize* the difference between statistical values of the previous imputation from the new augmented dataset. Therefore:

$$\mathcal{F} = |\mu_x - \mu_{x'}| + |\vartheta_x - \vartheta_{x'}| + \sum_{h=1}^{H} \frac{|\rho(h) - \rho'(h)|}{H}, \tag{5}$$

where μ_x, ϑ_x and $\rho(h)$ are respectively the mean, variance and auto-correlation of the time series being regressed and $\mu_{x'}$, $\vartheta_{x'}$ and $\rho'(h)$ are respectively the mean, variance and auto-correlation of the previously augmented data.

The auto-correlation is computed based on a sample located at distance h of a missing value, therefore the fitness is calculated taking into consideration the average of the auto-correlation of the H closest examples to the unknown data, where H is the maximum lag of the auto-correlation and is equal to 7, an arbitrary chosen value.

5.2. Pseudo-code

Algorithm 1 describes the steps of the GPImpute algorithm. At first, a temporary dataset X_{tmp} is created by replacing each missing value by the valid value closest to it. Since the auto-covariance function cannot deal with unknown data, it is not possible to compute the auto-correlation. Thus, if an example t is missing, it is replaced by an example $t + k$ where $k \in \mathbb{Z}$ and it must be the smallest integer possible. This approach is used to avoid breaks in local variations. Considering that the difference among adjacent observations are small, smooth fluctuations are maintained by choosing a neighbor sample. However, once a gap is too large, this heuristic is no longer valid, hence a more general method must be applied.

The second step is to find every attribute with missing values and apply the genetic programming algorithm, to build a regression function that represents the missing values distribution of the time series being analysed. The initial population is created randomly by *ramped half-and-half* tree construction method and the individuals are optimized generation to generation. Moreover, for each attribute x being regressed, each individual creates a new time series x' by replacing its missing values with the values calculated by the individual program, using X_{tmp} to get the unknown values of other attributes.

For instance, if a time series x_d is being regressed and the program that an individual represents is $x_{d+1} \times 0.75 + x_{d+2}$, to guarantee that every value of x_{d+1} and x_{d+2} is valid, the temporary dataset previously imputed X_{tmp} is used. Later, the resulted time series x' is compared to itself before any imputation (x_o). However, in x_o the examples with missing values are deleted, so the known statistical properties are maintained and no bias is added. Then, the regression function is evaluated according to the fitness given by Equation 5, comparing x' to x_o. Finally, at the end of every generation, the best individual is chosen. This process goes on until the fitness value is equal to zero or until the maximum number of generations is reached.

Algorithm 1: GPImpute Pseudo-code

```
Input  : Dataset with missing values; GPImpute params
Output: Augmented dataset; regression functions

1  $X_{mv} \leftarrow$ dataset with missing values
2  $X_{tmp} \leftarrow$ dataset imputed with missing value neighbors
3  for each attribute $a \in X_{mv}$ do
4      if $a$ has missing values then
5          Initialize GP individuals
6          Evaluate population using $X_{tmp}$ as testing set
7          while $g < maxGenerations$ or $fitness > 0$ do
8              Apply genetic operators
9              Evaluate individuals using $X_{tmp}$ as testing set
10             $f_c \leftarrow$ best individual regression function
11         for each example $x \in a$ do
12             if $x$ is missing then
13                 Apply $f_c$ to regress $x$ in $X_{mv}$ and $X_{tmp}$
```

Once the genetic programming algorithm finishes, the best individual is returned and it represents the regression function that best fits the time series. After it is found, the function is applied to estimate the missing values in the attribute being regressed. This process is repeated until there are no missing data left. It is worth mentioning that the other attributes in the database are used as parameters for regressing the time series function because it may find interrelations between different time series representing an unique phenomenon, making it possible for the GPImpute to explore correlations in a multivariate time series database.

6. Experiments

The experiments were conducted using seven different databases with three different proportions of missing data, 5%, 10% and 30%. Due to the fact that a Genetic Programming algorithm is a stochastic process, each case study was run 10 times. The datasets used are specified in Table 2, where five of them were picked from the UCI repository [Frank and Asuncion 2010], while the last two were selected from the KEEL repository [Alcalá et al. 2010]. The missing values were artificially introduced at random, with all categorical attributes were removed and a new feature indicating the time index was added when it was not present in the database already.

The GPImpute evaluated a population of 2048 individuals for 128 generations for every attribute with gaps in a database. Each feature not being regressed can be considered as a parameter for the equation to be built, what enables the algorithm to consider correlations among different time series. Moreover, it is worth mentioning that, since the first attribute of the studied database

Dataset	Acronym	# Attributes	# Instances	Application
ADL Climb Stairs	ACS	3	270	Activities of Daily Living
ADL Comb Hair	ACH	3	833	Activities of Daily Living
ADL Brush Teeth	ABT	3	3167	Activities of Daily Living
Indoor Movement	IM	4	27	Ambient Assistant
Istanbul Stock	IS	9	536	Finance
NN5 Complete 110	NN5	5	787	Finance
NNGC1 D1 V1 002	NNG	5	1175	Transportation

Table 2: Datasets used to test the GPImpute Algorithm.

represents the time, it can be considered to build the individuals equations. This approach adds more flexibility to the algorithm enabling it to deal with distributions not equally spaced in time.

Figure 2 illustrates an example of an individual built for regressing the second time series in *ADL Comb Hair* dataset. This individual represents the program $\frac{x[3]_t+x[1]_t}{0.739/0.391}$ and this equation estimates the value of every missing value in the time series represented by attribute 2, which means that $x[2]_t = \frac{x[3]_t+x[1]_t}{0.739} \times 0.391$ at a time t.

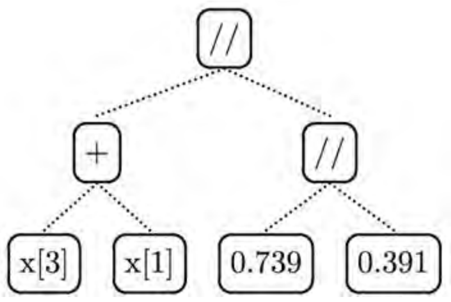


Figure 2: Regression Tree for Attribute 2 of ADL Comb Hair dataset.

It is worth mentioning that, for building this regression function, GPImpute used a correlation between attributes 1 and 3 ($X[1]$ and $X[3]$). This characteristic is a special property of the developed algorithm that enables it to find relationships among different multivariate datasets, what can be used by data analysts to extract knowledge about the missing pattern of the data distribution.

Finally, in order to evaluate the proposed algorithm, it was compared to three imputation methods available in KEEL software [Alcalá et al. 2010]. This algorithms are not the state of art methods used for time series imputation, however they represent three different methodologies commonly used for imputing data: clustering, mean related and neighbors related. The objective of this work is not to develop a competitive algorithm for time series imputation yet, but to code a viable alternative that generates a comprehensive model that can be used for knowledge extraction. In addition, the authors intend to improve the present research in the future to make it an efficient tool to be used in the area.

The methods to what GPImpute is compared are:

- **EC**: groups the samples into clusters based on their interdependence and extracts the pattern of each one for estimating the missing data [Wong and Chiu 1987].
- **MC**: replaces the missing values of a certain feature by the mean value of this feature. For categorical attributes, the missing data is replaced by the mode value [Batista and Monard 2003].
- **KMI**: divides the dataset into clusters based on the data similarities, aiming to minimize the intra-cluster dissimilarity, what is calculated by the differences between the samples and the

mean value of the cluster. The samples inside a cluster are considered neighbors and the nearest neighbor replaces the missing value [Li et al. 2004].

7. Results

To evaluate the performance of the proposed method, the GPImpute was compared to three well known algorithms for imputing data: EC, MC and KMI. The evaluation metrics are two: RMSE, which indicates how far from the real value the imputation was; and the difference on the auto-correlation at lag 7, which consists on calculating the difference of the auto-correlation of the original dataset and the augmented one. We have chosen to define the auto-correlation of the entire database the sum of the auto-correlation for every time series divided by the number of time series, or in other words, the average of the auto-correlation for all attributes. Then the absolute difference of the value obtained for the original dataset and the augmented one is taken. Considering that the auto-correlation value must lie in between +1 and -1, it can be said that the worst case possible is when the absolute difference is 2, e.g. (-1 - (+1)); and the best scenario is when this difference is 0, which means that the original properties of the time series were maintained.

Tables 3, 4 and 5 show the NRMSE (normalized RMSE) results for each database tested with 5%, 10% and 30% of missing values respectively, as well as a rank for the Friedman Test, a non-parametric statistical test that better demonstrates the difference of the performance of each method. The RMSE was normalized in order to perform statistical tests and to have a clearer scenario of the results obtained. The last line of each table shows the statistical results according to the *Holm/Shaffer* procedure at an $\alpha = 0.10$, a test that rejects false positive hypothesis. $\{A\} \succ \{B, C\}$ indicates that the method A is statistically better than the methods B and C.

Analysing the results, it can be noticed that, for all three proportions of missing data studied, the KMI algorithm outperformed GPImpute. A result that can be better evidenced by the Friedman Rank, which ranks the evaluated algorithms; the one with the lower value is the best algorithm analysed. In addition to that, it is also possible to note that GPImpute outperformed the algorithms EC and MC. However, according to *Holm/Shaffer* test, only comparisons between KMI, EC, MC and GP_μ are valid for datasets with 5% of missing values, and comparisons between KMI and GP_μ are valid for datasets with 10% and 30% of missing data.

Datasets	EC	KMI	MC	GPI_{Best}	GPI_μ
ABT	0.765	**1.000**	0.449	0.270	0.000
ACS	0.216	**1.000**	0.637	0.540	0.000
ACH	0.665	**1.000**	0.000	0.890	0.137
IM	0.000	**1.000**	0.657	0.795	0.689
IS	0.838	**1.000**	0.991	0.352	0.000
NN5	0.388	0.813	0.000	**1.000**	0.892
NNG	0.167	**1.000**	0.000	0.905	0.834
F. Rank	**3.571**	**1.285**	**3.714**	**2.571**	**3.857**

$\{KMI\} \succ \{EC, MC, GP_\mu\}$

Table 3: NRMSE and Friedman rank results for GPImpute algorithm for datasets with 5% of missing values.

Table 6 shows the difference of the auto-correlation of the original dataset and the augmented one. The results obtained show that, the larger the number of missing values in the dataset is, the better the results of GPImpute for auto-correlation evaluation metric are. What is an odd result since usually, the opposite happens.

Based on the results obtained from both evaluation metrics, some hypothesis can be pointed out:

- **The approach used for creating the X_{tmp} is inefficient:** to replace the missing values by its valid neighbor is a valid approach only for small gaps. For other cases it creates a constant

Datasets	EC	KMI	MC	GPI_{Best}	GPI_{μ}
ABT	0.762	**1.000**	0.388	0.350	0.000
ACS	0.384	0.481	**1.000**	0.789	0.000
ACH	0.507	**1.000**	0.034	0.996	0.000
IM	0.662	**1.000**	0.000	0.333	0.264
IS	0.780	0.998	**1.000**	0.380	0.000
NN5	0.414	0.998	0.000	**1.000**	0.911
NNG	0.000	0.562	0.143	**1.000**	0.887
F. Rank	**3.285**	**1.857**	**3.285**	**2.428**	**4.142**

$\{KMI\} \succ \{GPI_{\mu}\}$

Table 4: NRMSE and Friedman rank results for GPImpute algorithm for datasets with 10% of missing values.

Datasets	EC	KMI	MC	GPI_{Best}	GPI_{μ}
ABT	0.776	**1.000**	0.319	0.105	0.000
ACS	0.249	**1.000**	0.700	0.041	0.000
ACH	0.653	0.992	0.097	**1.000**	0.000
IM	0.949	**1.000**	0.909	0.918	0.000
IS	0.952	**1.000**	0.981	0.740	0.000
NN5	0.249	0.764	0.000	**1.000**	0.852
NNG	0.000	**1.000**	0.235	0.919	0.859
F. Rank	**3.142**	**1.142**	**3.428**	**2.714**	**4.285**

$\{KMI\} \succ \{GPI_{\mu}\}$

Table 5: NRMSE and Friedman rank results for GPImpute algorithm for datasets with 30% of missing values.

segment, what does not correspond to one of the basics aspects of a time series: smooth variations among samples.

- **The terminals chosen are not sufficient:** features not being regressed are important terminals because they allow the GPImpute to find correlations among multivariate time series in a time t, however past values (values in a time $t - k$) must be considered too.

- **The search space is very large:** due to the fact that a genetic programming algorithm is a stochastic process, it is possible to find several solutions for the problem. This fact can be evidenced by the large difference on the results obtained for GPI_{Best} and GPI_{μ}. Maybe, a clustering approach can be used to reduce the search space and make the regression functions less complex.

- **The time series might not be stationary:** statistical values such as mean, variance and auto-correlation can change over time. To build an unique regression function that defines an entire time series might introduce bias to the distribution. An algorithm that analyses these variations and builds a regression function for segments of a time series may be more efficient.

- **The RMSE is not a good evaluation metric:** due to the fact that in a real application the real value of the missing data is unknown, the RMSE may not reflect well the performance of an imputation algorithm. Therefore, depending on the application studied, other approaches must be taken into consideration. For time series applications it can be the difference of the auto-correlation values, which evaluates if the original statistical properties of the distribution was maintained. Hence, this evaluation metric is more relevant than the other one used in this article.

Dataset	EC	KMI	MC	GPI_{Best}	GPI_{μ}
			5%		
ABT	**0.013**	0.021	0.042	0.042	0.557
ACS	0.024	**0.008**	0.012	0.019	0.030
ACH	0.022	**0.010**	0.039	0.024	0.037
IM	0.045	**0.032**	0.089	0.052	0.056
IS	0.005	**0.001**	0.002	0.008	0.012
NN5	0.032	0.024	0.054	**0.000**	0.002
NNG	0.037	0.021	0.048	**0.000**	0.002
			10%		
ABT	**0.028**	0.032	0.065	0.088	0.979
ACS	0.058	**0.012**	0.018	0.023	0.042
ACH	0.068	**0.020**	0.093	0.047	0.084
IM	0.092	0.064	0.036	**0.027**	0.038
IS	0.005	0.006	0.005	**0.000**	0.006
NN5	0.047	0.019	0.085	**0.001**	0.002
NNG	0.094	0.002	0.084	**0.000**	0.001
			30%		
ABT	**0.100**	0.084	0.161	0.151	1.878
ACS	0.099	**0.045**	0.089	0.065	0.078
ACH	0.152	0.084	0.254	**0.070**	0.211
IM	0.027	0.016	0.086	**0.007**	0.048
IS	0.006	0.006	**0.002**	**0.002**	0.024
NN5	0.169	0.112	0.183	**0.008**	0.019
NNG	0.346	0.024	0.288	**0.000**	0.012

Table 6: Auto-correlation results for GPImpute algorithm for datasets with 5%, 10% and 30% of missing values.

Notwithstanding the low performance of our method in terms of RMSE and in terms of auto-correlation for datasets with 5% of missing values, the authors believe that the developed method is promising for two main reasons: (i) it builds an interpretable model and (ii), it was capable to maintain the statistical properties of most cases studied. Moreover, by remodeling the algorithm developed in order to explore each of hypothesis made, the GPImpute may become a more viable methodology for estimating missing values in multivariate time series data.

8. Conclusion

In this paper, it was proposed a Genetic Programming algorithm to estimate missing values in multivariate time series, called GPImpute. The method is based on two concepts: 1) time series are a stochastic process where features such as mean, variance and auto-correlation are vital statistics to be considered when treating missing values, for maintaining its original properties; and 2) a Genetic Programming algorithm is able to efficiently regress a function based on example data. As a result, GPImpute builds an comprehensive model that, in addition to impute data, can also be used to analyse patterns.

The GPImpute was tested on seven datasets of different segments, sizes and proportion of missing values, plus it was compared against three well known algorithms available in the KEEL software: KMI, EC and MC. The proposed method outperforms EC and MC methods but falls behind KMI in terms of RMSE. However, it shows promising results when evaluating if it is capable to maintain the statistical properties of the distribution. In addition to that, the GPImpute is the only one that provides a comprehensive model, what is important for forecasting and control applications and also allows data analysts to better understand the missing values characteristics.

In general, the results obtained evidence a failure in our modelling phase. Although, by analysing them, new hypothesis were pointed out, which, if taken into consideration, can improve the algorithm's performance. For future research, we intend to replace the approach chosen to create the temporary set by a polynomial interpolation method, which we believe, will improve the accuracy of the initial imputation, so it can be optimized by the Genetic Programming algorithm. Moreover, we intend to consider the past values of a time series in the regression functions, and to apply a clustering algorithm to better define the search space for building each regression function.

Acknowledgements

The authors are thankful to the Committee for Postgraduate Courses in Higher Education (CAPES) and the National Council for Scientific and Technological Development (CNPq) by the financial supporting of this research.

References

Alcalá, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., and Herrera, F. (2010). KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, 17(2-3), pp. 255-287.

Batista, G. E., and Monard, M. C. (2003). An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence*, 17(5-6), pp. 519-533.

Box, G. E., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (2008). Time series analysis: forecasting and control. *John Wiley & Sons*.

Cai, Y., Tong, H., Fan, W., Ji, P., and He, Q. (2015). Facets: Fast comprehensive mining of coevolving high-order time series. *In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* pp. 79-88. ACM.

Dai, L., Zanobetti, A., Koutrakis, P., and Schwartz, J. D. (2014). Associations of fine particulate matter species with mortality in the United States: a multicity time-series analysis. *Environmental Health Perspectives (Online)*, 122(8), 837.

Eiben, A. E., and Smith, J. E. (2003). Introduction to evolutionary computing (Vol. 53). *Heidelberg: springer*.

Figueroa García, J. C., Kalenatic, D., and Lopez Bello, C. A. (2010). An evolutionary approach for imputing missing data in time series. *Journal of Circuits, Systems, and Computers* pp. 107-121.

Frank, A., and Asuncion, A. (2010). *UCI machine learning repository*.

Freitas, A. A. (2013). Data mining and knowledge discovery with evolutionary algorithms. *Springer Science & Business Media*.

García, J. C. F., Kalenatic, D., and Bello, C. A. L. (2011). Missing data imputation in multivariate data by evolutionary algorithms. *Computers in Human Behavior*, 27(5), pp. 1468-1474.

Honaker, J., and King, G. (2010). What to do about missing values in timeseries crosssection data. *American Journal of Political Science*, 54(2), pp. 561-581.

Junger, W. L., and de Leon, A. P. (2015). Imputation of missing data in time series for air pollutants. *Atmospheric Environment*, 102, pp. 96-104.

Li, D., Deogun, J., Spaulding, W., and Shuart, B. (2004). Towards missing data imputation: a study of fuzzy k-means clustering method. In International Conference on Rough Sets and Current Trends in Computing pp. 573-579. *Springer Berlin Heidelberg*.

Liu, S., and Molenaar, P. C. (2014). iVAR: A program for imputing missing data in multivariate time series using vector autoregressive models. *Behavior research methods*, 46(4), pp. 1138-1148.

Lobato, F., Sales, C., Araújo, I., Tadaiesky, V., Dias, L., Ramos, L., and Santana, Á. (2015a). Multi-objective genetic algorithm for missing data imputation. *Pattern Recognition Letters*.

Lobato, F. M., Tadaiesky, V. W., Araújo, I. M., and de Santana, Á. L. (2015b). An Evolutionary Missing Data Imputation Method for Pattern Classification. *In Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM.

Luke, S., Panait, L., Balan, G., Paus, S., Skolicki, Z., Popovici, E., ... and Chircop, A. (2004). A java-based evolutionary computation research system. Web page. http://cs.gmu.edu/eclab/projects/ecj. Accessed in 2016-01-28.

Patil, D. V., and Bichkar, R. S. (2010). Multiple imputation of missing data with genetic algorithm based techniques. *IJCA Special Issue on Evolutionary Computation for Optimization Techniques*.

Poli, R., Langdon, W. B., McPhee, N. F., and Koza, J. R. (2008). A field guide to genetic programming. Lulu. com.

Shumway, R. H., and Stoffer, D. S. (2010). Time series analysis and its applications: with R examples. *Springer Science & Business Media.*

Tran, C. T., Zhang, M., and Andreae, P. (2015). Multiple imputation for missing data using genetic programming. *In Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation* pp. 583-590. ACM.

Wellenzohn, K., Mitterer, H., Gamper, J., B ohlen, M. H., and Khayati, M. (2014). Missing Value Imputation in Time Series Using Top-k Case Matching. *In Grundlagen von Datenbanken*.

Wolfgang, B., Nordin, P., Keller, R., and Francone, F. (1998). Genetic programming: An Introduction: on the automatic evolution of computer programs and its applications. *Co Published by Morgan Kaufmann & Verlag*.

Wong, A. K., and Chiu, D. K. (1987). Synthesizing statistical knowledge from incomplete mixed-mode data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6), pp. 796-805.