A Vectorial Approach to Genetic Programming

A Vectorial Approach to Genetic Programming

Irene Azzali[1 [0000−0002−5808−7044]] ✉, Leonardo Vanneschi[2 [0000−0003−4732−3328]], Sara Silva[3,4 [0000−−0001−−8223−−4799]], Illya Bakurov[2 [0000−−0002−−6458−−942X]] and Mario Giacobini[1 [0000−0002−7647−5649]]

[1] DAMU - Data Analysis and Modeling Unit, Department of Veterinary Sciences, University of Torino, Italy
{irene.azzali,mario.giacobini}@unito.it
[2] NOVA Information Management School (NOVA IMS), Universidade Nova de Lisboa, Campus de Campolide, 1070-312 Lisboa, Portugal
{lvannesc,ibakurov}@novaims.unl.pt
[3] LASIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal
sara@fc.ul.pt
[4] BioISI – Biosystems & Integrative Sciences Institute, Faculdade de Ciências, Universidade de Lisboa, Portugal

Abstract. Among the various typologies of problems to which Genetic Programming (GP) has been applied since its origins, symbolic regression is one of the most popular. A common situation consists in the prediction of a target time series based on scalar features and other time series variables collected from multiple subjects. To manage this problem with GP data needs a *panel* representation where each observation corresponds to a collection on a subject at a precise time instant. However, representing data in this form may imply a loss of information: for instance, the algorithm may not be able to recognize observations belonging to the same subject and their recording order. To maintain the source of knowledge supplied by ordered sequences as time series, we propose a new approach to GP that keeps instances of the same observation together in a vector, introducing vectorial variables as terminals. This new representation allows aggregate functions in the primitive GP set, included with the purpose of describing the behaviour of vectorial variables. In this work, we perform a comparative analysis of vectorial GP (VE-GP) against standard GP (ST-GP). Experiments are conducted on different benchmark problems to highlight the advantages of this new approach.

Keywords: Genetic Programming · Vector-based representation · Panel Data regression.

1 Introduction

Among the several existing typologies of problems to which Genetic Programming (GP) [1] can be applied, symbolic regression is undoubtedly one of the most popular. The objective of symbolic regression is to find a function that describes the relationship between inputs and corresponding outputs, developing a

model that can be used to make predictions on new inputs. Of great importance, belonging to the family of symbolic regression, is *panel data forecasting*.

A panel dataset is a collection of observations for multiple subjects at different equal-spaced time intervals [2]. Therefore, if the independent variables among the M observations measured are $X_1^i, \ldots, X_N^i$ where $i = 1, \ldots, M$ and $X_K^i, \ldots, X_N^i$ change in time ($X_{j_t}^i$ with $i = 1, \ldots, M$, $j = K, \ldots, N$ and $t = 1, \ldots, T$ denoting time series variables) and Y is a dependent variable (Y_t^i with $t = 1, \ldots, T$ denoting a target time series variable), we can express the dataset as

$$\{X_1^i, \ldots, X_{K-1}^i, X_{K_1}^i, \ldots, X_{K_T}^i, \ldots, X_{N_1}^i, \ldots, X_{N_T}^i, Y_1^i, \ldots, Y_T^i\}$$

where i= 1,. . . ,M refers to the subject being observed. The interest of panel data regression lies in predicting dependent variables which are hard to measure. To clarify, let us consider the example panel reported in Table 1. In this example,

Table 1. Example standard panel dataset.

Person ID	Age	Sex	Year	*Income*
1	27	1	2015	1600
1	28	1	2016	1500
2	42	2	2015	1900
2	43	2	2016	2000
2	44	2	2017	2100
3	34	1	2015	3300

individual characteristics are collected for different persons and years in order to predict the income. The standard GP approach can be easily applied to panel data regression; however, there can be a potential disadvantage. Data instances (fitness cases) are treated independently. Therefore the algorithm is not able to recognize that two (as in lines 1 and 2 in Table 1), or more, observations belong to the same individual. This situation may result in a loss of knowledge regarding the time series, that may instead have been useful to effectively model the target.

The idea behind this work is to design a novel GP system, that we call *Vectorial GP* (VE-GP), able to exploit the source of information provided by the additional dimension of time of panel datasets. To make the algorithm consider the whole time-series, we aggregate related data instances referring to the same entity in a vectorial representation, so that variables that change in time become *vectorial variables*. Therefore, the panel dataset represented in Table 1 is transformed into the representation reported in Table 2. In this configuration, a GP tree can be composed either by scalar terminals (to represent features such as Sex and Person ID) or by vectorial terminals (for instance to represent

Table 2. Example vectorial panel dataset.

Person ID	Ages	Sex	Years	*Incomes*
1	[27,28]	1	[2015,2016]	[1600,1500]
2	[42,43,44]	2	[2015,2016,2017]	[1900,2000,2100]
3	34	1	2015	3300

Ages and Years). Moreover, the technique we propose adds new functions to the primitive set, defined with the purpose of describing the behaviour of temporal, or more generally vectorial, variables. From now on, we use the term *time series* to indicate an observation of a vectorial variable, also called *time series variable*. Therefore, a time series is a sequence of recorded values, belonging to one entity and represented as a vector.

The paper is organized as follows: in Section 2, we present previous and related work and we motivate the novelty of our approach. In Section 3, we describe VE-GP focusing on the new techniques and approaches we propose in the different blocks of a GP structure. Test problems and experimental setting used to explore the performance of VE-GP are presented in Section 4. Sections 5 and 6 describe the result obtained, comparing VE-GP with the standard GP and discuss the possible limitations of the new approach. Finally, in Section 7, we conclude with final considerations, and we give ideas for possible future works.

2 Previous and Related Work

Working with time series in GP has always been a challenging problem due to the inherent difficulty of handling this type of data. Common strategies include feature extractions to reduce the series into scalar features [3] or element by element treatment, where each entry of the series is an independent terminal [4]. However, some previous works explored the idea of keeping the native data type of time series, the vector. Holladay et al. [5] introduced a vector-based GP to predict the feature vector of fixed length signals. In this paper, vectors were possible inputs rather than scalars, and the primitive set included domain dependent functions that act on both of them. A more recent work of Bartashevich et al. [6] allows vectors as primitives and include vectorial functions such as the cross and dot product to build GP individuals. Other approaches have been proposed to preserve the ordered essence of time series such as Vera et al. [7]. In this latter work the authors investigated the serial processing of data where the time sequence is presented to the algorithm in series so that the elements of a sequence are processed in the same order as they are recorded.

Starting from the idea of [5] we move further to provide a more exhaustive vector-based GP. Our VE-GP is less problem-specific and aims at providing an algorithm able to deal with any naturally ordered variable of any length. Moreover, in VE-GP we have included new structures that advance the search

space of the considered problems. VE-GP elevates the capabilities of previously proposed vector-based approaches and provides a more sophisticated technique.

3 Vectorial GP

VE-GP is built on top of the GPLAB toolbox [8]. GPLAB includes most of the traditional features usually found in many GP systems. We have chosen it because its highly modular structure makes it a particularly versatile, generalist and easily extendable tool, highly suited for testing new elements and techniques. Moreover, it is written in MATLAB, which provides a particularly appropriate environment to manage vectors. In the following paragraphs we describe the primitive set and other particular elements of VE-GP.

Functions of Arity One Since VE-GP is specific for time series variables, we have integrated the set of classical arity one functions with aggregate functions. Standard aggregate functions collapse the whole time series variable into a single value of more significant meaning. They can be included in the primitive set when we deal with time series prediction based on past time series variables. We even face problems where the time series target flows simultaneously to the time series predictors, which means that the time instants, corresponding to the entries of a vector, are the same for both the target and the predictors. Therefore, specially meant for this latter problem, we have added cumulative aggregate functions. These operators applied on a time series return a vector whose entries are the aggregate values of only previous time values. These versions of the aggregate function, the standard and the cumulative ones, allow GP to foresee any kind of time series, from the ones that take place during the recording of data to the future ones. All the arity one functions can be also easily applied to scalars, considering them as a vector of 1×1 dimension. The primitives of arity one used by VE-GP are described in Table 3.

Functions of Arity Two Concerning arity two we have included new functions inspired by classical vector operations. These functions can manage vectors of different lengths completing the shortest one with the null-element of the function involved. In the case of a scalar and a vector as inputs we have provided a specific evaluation in order not to consider scalars as 1×1 vectors. The primitives of arity two used by VE-GP are described in Table 4. The functions `VSUMW`, `V_W`, `VprW` and `VdivW` are called standard arity two functions.

Parametric Aggregate Functions We have introduced parametric aggregate functions that apply the referring aggregate function only to the values belonging to the time window described by parameters. Regarding standard aggregate functions, the parameters p and q define respectively the initial and final position of the range to be considered. Therefore the standard aggregate function is applied to the input values of position $p, \ldots, q-1, q$. To have an admissible range

Table 3. Description of functions of arity one. The columns represent the primitive function (first column), MATLAB name (second column) and the outcome of the function (third column) applied on a given vector v (in this example, v=[1, 2.5, 4.3, 0.7]).

Primitive function (pf)	**MATLAB name**	**pf(v)**
Mean	`V_mean`	`V_mean`$([1, 2.5, 4.3, 0.7]) = 2.1$
Max, Min	`V_max`, `V_min`	`V_max`$([1, 2.5, 4.3, 0.7]) = 4.3$ `V_min`$([1, 2.5, 4.3, 0.7]) = 0.7$
Sum	`V_sum`	`V_sum`$([1, 2.5, 4.3, 0.7]) = 8.5$
Mode	`V_mode`	`V_mode`$([1, 2.5, 4.3, 0.7]) = 0.7$
Length	`V_length`	`V_length`$([1, 2.5, 4.3, 0.7]) = 4$
2-Norm	`V_2norm`	`V_2norm`$([1, 2.5, 4.3, 0.7]) = 5.1$
Cumulative mean	`C_mean`	`C_mean`$([1, 2.5, 4.3, 0.7]) = [1, 1.8, 2.6, 2.1]$
Cumulative sum	`C_sum`	`C_sum`$([1, 2.5, 4.3, 0.7]) = [1, 3.5, 7.8, 8.5]$
Cumulative max, min	`C_max`, `C_min`	`C_max`$([1, 2.5, 4.3, 0.7]) = [1, 2.5, 4.3, 4.3]$ `C_min`$([1, 2.5, 4.3, 0.7]) = [1, 1, 1, 0.7]$
Exp, Log, Cos, Sin, Abs, Square, Cube, Sqrt	`V_exp`, `V_log`*, `V_cos`, `V_sin`, `V_abs`, `V_2`, `V_3`, `V_sqrt`* *(protected version as [1])	`V_exp`$([1, 2.5, 4.3, 0.7]) = [2.7, 12.2, 73.7, 2.0]$ `V_log`$([1, 2.5, 4.3, 0.7]) = [0, 0.9, 1.5, -0.4]$ `V_cos`$([1, 2.5, 4.3, 0.7]) = [0.5, -0.8, -0.4, 0.8]$ `V_sin`$([1, 2.5, 4.3, 0.7]) = [0.8, 0.6, -0.9, 0.6]$ `V_abs`$([1, 2.5, 4.3, 0.7]) = [1, 2.5, 4.3, 0.7]$ `V_2`$([1, 2.5, 4.3, 0.7]) = [1, 6.3, 18.5, 0.5]$ `V_3`$([1, 2.5, 4.3, 0.7]) = [1, 15.6, 79.5, 0.3]$ `V_sqrt`$([1, 2.5, 4.3, 0.7]) = [1, 1.6, 2.1, 0.8]$

$p < q$. Concerning cumulative aggregate functions, we remind that the output is a vector. The $i-$ith entry of the output depends on the values belonging to the window described by parameters p and q. In this case, p defines how far to look back from the i position determining the initial value of the range, while q defines how many values to consider. Thus, the $i-$ith entry of the output is the aggregate function applied on input values of position $i-(p-1), \ldots, i-(p-1)+q-1$. To have admissible range in this case $p > q$. The primitives of the parametric aggregate function used by VE-GP are described in Table 5.

It is noteworthy that many of the new functions are not replicable by the standard GP.

Table 4. Description of functions of arity two. The columns represent the primitive function (first column), MATLAB name (second column), the outcome of the function (third column) applied on a given scalar s and vector v1 and the outcome of the function (fourth column) applied on two vectors v1 and v2 (in this example, s=0.2, v1=[1, 2.5, 4.3], v2=[0.1, 3.2, 4, 1.1]).

Primitive function (pf)	**MATLAB name**	**pf(s,v1)**	**pf(v1,v2)**
Element-wise sum	`VSUMW`	`VSUMW`$(0.2, [1, 2.5, 4.3]) = [1.2, 2.7, 4.5]$	`VSUMW`$([1, 2.5, 4.3], [0.1, 3.2, 4, 1.1]) = [1.1, 5.7, 8.3, 1.1]$
Element-wise	`V_W`	`V_W`$(0.2, [1, 2.5, 4.3]) = [-0.8, -2.3, -4.1]$	`V_W`$([1, 2.5, 4.3], [0.1, 3.2, 4, 1.1]) = [0.9, -0.7, 0.3 - 1.1]$
Element-wise product	`VprW`	`VprW`$(0.2, [1, 2.5, 4.3]) = [0.2, 0.5, 0.9]$	`VprW`$([1, 2.5, 4.3], [0.1, 3.2, 4, 1.1]) = [0.1, 8, 17.2, 1.1]$
Scalar	`VscalprW`	`VscalprW`$(0.2, [1, 2.5, 4.3]) = 1.6$	`VscalprW`$([1, 2.5, 4.3], [0.1, 3.2, 4, 1.1]) = 26.4$
Element-wise division	`VdivW`* *(protected version as [1])	`VdivW`$(0.2, [1, 2.5, 4.3]) = [0.2, 0.08, 0.05]$	`VdivW`$([1, 2.5, 4.3], [0.1, 3.2, 4, 1.1]) = [10, 0.8, 1.1, 0.9]$
Scalar Division	`VscaldivW`* *(protected version as [1])	`VscaldivW`$(0.2, [1, 2.5, 4.3]) = 0.3$	`VscaldivW`$([1, 2.5, 4.3], [0.1, 3.2, 4, 1.1]) = 12.8$

Initialization Given the new representation in VE-GP, several challenges arise. Firstly, a big number of scalar inputs can cause a poor initial representation of new functions and terminals, as such barely used during the evolutionary process. Secondly, it is possible to obtain final solutions whose output is a scalar and not a vector because many of the integrated functions collapse a vector into a scalar. We have designed a different initialization strategy which releases unique and innovative characteristics of VE-GP during the evolution. The strategy is resumed in the procedure described in Figure 1.

The motivation behind the second rule is to ensure a representative amount of trees in the initial population whose output is not a scalar. Similarly, the third rule ensures trees where the new functions are meaningfully used. In our opinion, the initialization strategy that we propose does not introduce significant bias in the evolutionary process, contrarily, it aims to aid VE-GP to free its full potential during the evolution.

We have furthermore initialized the values of the parameters for the aggregate functions. Because a time series variable can have a different length among fitness cases, we have randomly set p and q between 1 and the maximum time series length for all the parametric aggregate functions.

Table 5. Description of parametric aggregate functions. The columns represent the primitive function (first column), MATLAB name (second column), the outcome of the function (third column) applied on a given vector. For standard functions p is the initial position while q is the final position of the range, instead for cumulative functions p is the number of backward steps to be made in order to determine the initial position of the range while q is the amplitude of the range (in this example, v=[1, 2.5, 4.3, 0.7, 1.6], (p,q)=(2,3) for standard functions, (p,q)=(3,2) for cumulative functions).

Primitive function (pf)	**MATLAB name**	**pf$_{p,q}$(v)**
Mean in [p,q]	V_mean$_{p,q}$	V_mean$_{2,3}([1, 2.5, 4.3, 0.7, 1.6]) = 3.4$
Max in [p,q], Min in [p,q]	V_max$_{p,q}$, V_min$_{p,q}$	V_max$_{2,3}([1, 2.5, 4.3, 0.7, 1.6]) = 4.3$ V_min$_{2,3}([1, 2.5, 4.3, 0.7, 1.6]) = 2.5$
Sum in [p,q]	V_sum$_{p,q}$	V_sum$_{2,3}([1, 2.5, 4.3, 0.7, 1.6]) = 6.8$
Cumulative mean in [p,q]	V_Cmean$_{p,q}$	V_Cmean$_{3,2}([1, 2.5, 4.3, 0.7, 1.6]) = [0, 1, 1.8, 3.4, 2.5]$
Cumulative max in [p,q], Cumulative min in [p,q]	V_Cmax$_{p,q}$, V_Cmin$_{p,q}$	V_Cmax$_{3,2}([1, 2.5, 4.3, 0.7, 1.6]) = [0, 1, 2.5, 4.3, 4.3]$ V_Cmin$_{3,2}([1, 2.5, 4.3, 0.7, 1.6]) = [0, 1, 1, 2.5, 0.7]$
Cumulative sum in [p,q]	V_Csum$_{p,q}$	V_Csum$_{3,2}([1, 2.5, 4.3, 0.7, 1.6]) = [0, 0, 3.5, 6.8, 5]$

Create an empty population P (the initial population) of size N.

1. Generate $n1$ trees in P using Ramped Half-and-Half initialization algorithm (RHH) [1];
2. Generate $n2$ trees in P using RHH such that each tree always generates an output which is a vector:
   (a) randomly generate a tree t by means of RHH;
   (b) randomly select a standard primitive function pf of arity two;
   (c) randomly select a vector-terminal v;
   (d) create the following tree, using post-fix notation, *(pf t v)*;
3. Generate $n3$ trees in P using RHH where aggregate primitive functions are forced to receive a vector-terminal as an input.

Fig. 1. Proposed initialization strategy.

Genetic Operators VE-GP includes a new type of mutation. Besides the classical one there is the mutation of aggregate function parameters. This operator allows parameters to evolve so that the most informative window where to apply the relative aggregate function is found. Firstly, the algorithm searches the tree for parametric aggregate functions and randomly selects one. Secondly, a random parameter is chosen and mutated according to the procedure reported in Table 6. Every time a genetic operator requires a new tree, parameters are set according to the initialization default values.

Table 6. Parameters mutation.

Standard aggregate functions parameters p,q	**Cumulative aggregate function parameters p,q**
– Random selection of p or q; – If p randomly change it from 1 to q; – If q randomly change it from p to the maximum time series length.	– Random selection of p or q; – If p randomly change it from 1 to the maximum time series length; – If q randomly change it from 1 to p.

4 Experiments

4.1 Benchmark Problems

We have tested the proposed VE-GP against a standard GP system (ST-GP) on four benchmark problems. To investigate the competitiveness of VE-GP we have chosen a first problem where the target does not involve the new primitive functions in order to see if VE-GP is penalized by having unnecessary functions and structures. Three more problems include some of the new functions in the target, and they are meant to show the performances of both algorithms considering that ST-GP can just try to approximate the new functions at its best.

Korns5 This benchmark problem is inspired by Korns problem number five for symbolic regression [9]. We have chosen to involve four variables of random numbers between -50 and 50 as the input, named $X1, X2, X3, X4$ respectively. Differently from the true Korns problem number five, the latter variable $X4$ for our experiment is a vector of length 10. The target expression is:

$$K5 = \texttt{VSUMW}(3.0, \texttt{VprW}(2.13, \texttt{V\_log}(X4))).$$

The dataset for VE-GP consists of 1000 instances, while for ST-GP it consists of 10000 instances because we have vertically untied the variable $X4$ and the target $K5$ to have the classical panel data representation.

Benchmark1 This benchmark problem is a new one that makes use of the aggregate functions implemented to produce the target. The vectorial dataset consists of 100 instances, where each instance is composed by four features: a random number between -10 and 10, another random number between -10 and 10, a vector of random numbers between 10 and 40 of length 10, and a vector of random numbers between -5 and 5 of length 10. Naming the variables in order as $X1, X2, X3$ and $X4$ the target reads as follows:

$$B1 = \texttt{V\_W}\Big(\texttt{VSUMW}(X4, \texttt{V\_mean}(X3)), \texttt{V\_W}(\texttt{VscalprW}(X3, X4), X2)\Big).$$

Conversely, for the scalar dataset, we have vertically untied the vectorial features so that it consists of 1000 instances.

Benchmark2 This benchmark problem involves the cumulative functions described in the previous section. The vectorial dataset is the same used for the previous benchmark B1, while the target is now

$$B2 = \texttt{VprW}\Big(\texttt{VdivW}(\texttt{VSUMW}(X3, X1), \texttt{V\_Cmean}(X4)), X2\Big).$$

Again for the scalar dataset we have vertically untied the vectorial variables.

Benchmark3 This benchmark problem includes parametric aggregate functions. Therefore the evolution of parameters is integrated in VE-GP. The variables involved are five: $X1$ is a vector of length 20 of random numbers between 10 and 30, $X2$ is a random number between 50 and 60, $X3$ is a random number between 5 and 10, $X4$ is a random number between -2 and 2, and $X5$ is a random number between 0 and 1. The target is:

$$B3 = \texttt{VSUMW}(\texttt{VprW}(\texttt{V\_Cmin}_{3,3}(X1), \texttt{VdivW}(X2, X3)), X4).$$

The dataset for VE-GP consists of 100 instances, while for ST-GP it consists of 1000 instances because, as for the other problems, we have vertically untied the vectorial variables.

4.2 Parameters and Statistical Test

The experimental parameters used in all the problems are provided in Tables 7 and 8. They were essentially the same for both ST-GP and VE-GP to facilitate the comparison between the techniques. We should remark that the choice of new terminal functions between cumulative or standard version depends on the chosen recording time of the time series involved. Fitness is calculated as the Root Mean Square Error (RMSE) between the output and the target. Since the output of trees built by VE-GP is supposed to be a vector, for this latter algorithm we have calculated the RMSE vertically disbanding both output and target; in this way the measures of fitness are ensured to be comparable between the two techniques. Moreover, when a VE-GP tree wrongly produces scalars as

an output, each scalar is replicated until the length of the corresponding target to make it a vector. We have decided to penalize these trees by multiplying their fitness for a huge constant (100).
We have tested both techniques on a total of 50 runs, each of which considers a different training and test data partition; from now on the term test set stands for unseen data. In particular, at the beginning of a VE-GP run, 70% of the instances are randomly selected as the training set, while the remaining ones form the test set. We have kept the same division for ST-GP with correspondence of the observations, so that time series are not split. We have performed a set of tests to analyse the statistical significance of the results. At first, the Kolmogorov-Smirnov test has shown that final fitness data are not normally distributed and hence we have opted for a rank-based statistic. Test decision for the null hypothesis of no difference in performance between ST-GP and VE-GP has been calculated with the Wilcoxon Rank Sum Test on the final test fitness. We have opted for it because it is a non parametric test and considers the median which is more robust than the mean to outliers. In order to quantify the assuming difference in performance between the two approaches, we have even used a Vargha Delaney A-test which is an index of effect size [10].

5 Results

In this section, we analyse the performance achieved by the two algorithms on the four problems. The evolution fitness plot (Figure 2) shows the best fitness in each generation for the training and the test set, median of 50 runs. Besides evolution plots, there are boxplots based on the test fitness at the end of evolution. The statistical test comparing final test fitness between both techniques can be found in Table 9. In this table, p is the p-value of the Wilcoxon test with a 5% level of significance. The term A represents the value of the Vargha Delaney A-test. The test returns a number between 0 and 1, representing the probability that a randomly selected observation from the first sample is bigger than a randomly selected observation from the second sample. In our specific case, the first sample is formed by the best fitness found by ST-GP while the second sample is composed of the best fitness found by VE-GP. It is important to remember that fitness is measured via RMSE, therefore, the lower it is, the better performance it means. Vargha and Delaney in [10] provided a suggested threshold for interpreting the size of the difference: 0.5 means no difference at all, up to 0.56 indicates a small difference, up to 0.64 indicates medium and anything over 0.71 is large. The same intervals apply below 0.5.

Firstly, if we consider the K5 benchmark, there is no significant disparity in performance between the algorithms. This confirms our expectation since the target of the problem does not involve the new functions; the difference between techniques, thus, it is just in data representation. The VE-GP algorithm moreover is not affected by unnecessary improvement of the initialization step and by the extension of the primitive set. Table 9 and Figure 3 show differently that VE-GP outperforms ST-GP for B1, B2, and B3. Moreover, Figure 2 reveals

Table 7. Standard GP parameters.

	K5	**B1**	**B2**	**B3**
Runs	50	50	50	50
Population	500	500	500	500
Generations	100	100	100	100
Training-Testing division	70%-30% of vectorial instances	70%-30% of vectorial instances	70%-30% of vectorial instances	70%-30% of vectorial instances
Genetic operators	Crossover, probability 0.9-Mutation, probability 0.1	Crossover, probability 0.9-Mutation, probability 0.1	Crossover, probability 0.9-Mutation, probability 0.1	Crossover, probability 0.9-Mutation, probability 0.1
Initialization	Ramped Half-and-Half [1], max depth 6	Ramped Half-and-Half [1], max depth 6	Ramped Half-and-Half [1], max depth 6	Ramped Half-and-Half [1], max depth 6
Functions set	plus, minus, times, protected div as [1]	plus, minus, times, protected div as [1]	plus, minus, times, protected div as [1]	plus, minus, times, protected div as [1]
Terminals set	Input variables, random numbers	Input variables, random numbers	Input variables, random numbers	Input variables, random numbers
Selection for reproduction	Lexicographic Parsimony Pressure [11], tournament size=10	Lexicographic Parsimony Pressure [11], tournament size=10	Lexicographic Parsimony Pressure [11], tournament size=10	Lexicographic Parsimony Pressure [11], tournament size=10
Elitism	Replication probability 0.1, best individual is kept	Replication probability 0.1, best individual is kept	Replication probability 0.1, best individual is kept	Replication probability 0.1, best individual is kept
Maximum depth	17	17	17	17

an increasing error for the ST-GP test set on both B1 and B2 problems which means that overfitting is occurring. Therefore ST-GP is not able to understand the underlying relationship between the data. This phenomenon does not happen to VE-GP that increases in fitness during generation for both training and test data. A notable observation that emerges from the B2 evolution plot is the growing amplitude of percentiles. This consideration stresses the fact that every time ST-GP tries to extract the implicit relationship between data it fails, remaining stuck to high error levels. Concerning B3, the difficulty of finding the correct window of time emerges even from VE-GP, where percentiles show the presence of runs not able to overcome ST-GP in 50 generations. Nevertheless, at the end of the evolutionary process, every VE-GP model outperforms the ST-GP ones that stabilize at high error suggesting the idea of no future improvements.

Table 8. Vectorial GP parameters.

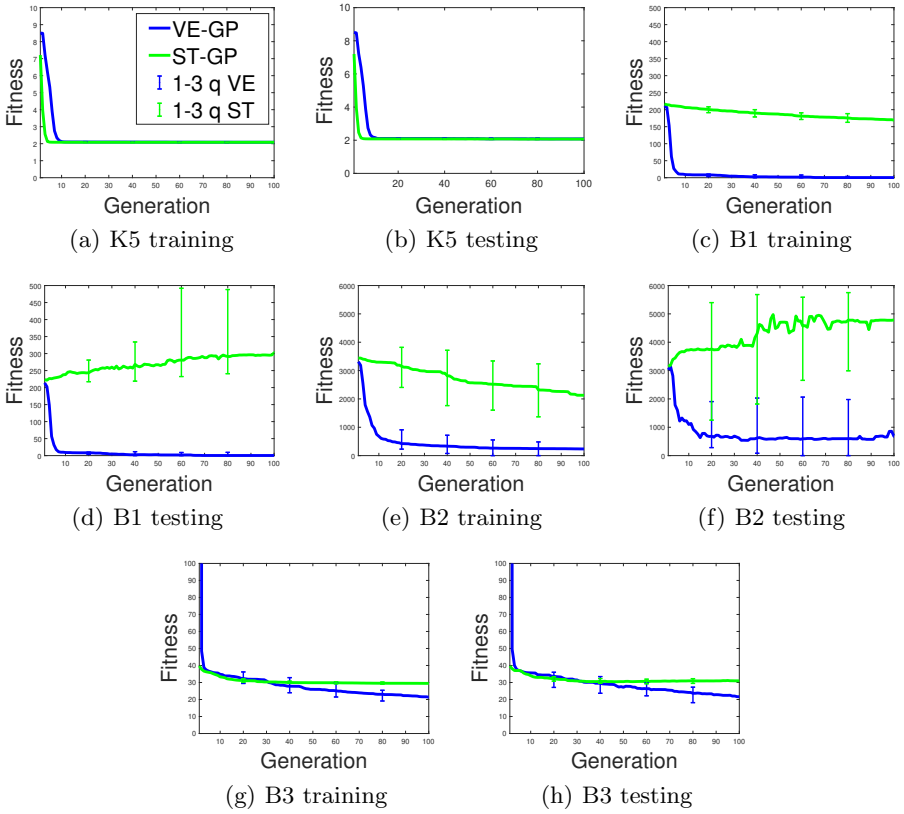
	K5	**B1**	**B2**	**B3**
Runs	50	50	50	50
Population	500	500	500	500
Generations	100	100	100	100
Training-Testing division	70%-30% of instances	70%-30% of instances	70%-30% of instances	70%-30% of instances
Genetic operators	Crossover, probability 0.9-Mutation, probability 0.1	Crossover, probability 0.9-Mutation, probability 0.1	Crossover, probability 0.9-Mutation, probability 0.1	Crossover, probability 0.5-Mutation, probability 0.1-Mutation of parameters, probability 0.4
Initialization	30% Ramped Half-and-Half [1], 70% Ramped Half-and-Half with forced initialization	30% Ramped Half-and-Half [1], 70% Ramped Half-and-Half with forced initialization	30% Ramped Half-and-Half [1], 70%Ramped Half-and-Half with forced initialization	30% Ramped Half-and-Half [1], 70%Ramped Half-and-Half with forced initialization
Functions set	`VSUMW, V_W, VprW, VdivW, V_mean, V_max, V_min,V_sum`	`VSUMW, V_W, VprW, VdivW, V_mean, V_max, V_min, V_sum VscalprW`	`VSUMW, V_W, VprW, VdivW, V_Cmean, V_Cmax, V_Cmin, V_Csum`	`VSUMW, V_W, VprW, VdivW, V_Cmeanpq, V_Cmaxpq, V_Cminpq, V_Csumpq`
Terminals set	Input variables, random numbers	Input variables, random numbers	Input variables, random numbers	Input variables, random numbers
Selection for reproduction	Lexicographic Parsimony Pressure [11], tournament size=10	Lexicographic Parsimony Pressure [11], tournament size=10	Lexicographic Parsimony Pressure [11], tournament size=10	Lexicographic Parsimony Pressure [11], tournament size=10
Elitism	Replication probability 0.1, best individual is kept	Replication probability 0.1, best individual is kept	Replication probability 0.1, best individual is kept	Replication probability 0.1, best individual is kept
Maximum depth	17	17	17	17

Table 9. Statistical results of final test fitness comparison.

K5	**B1**	**B2**	**B3**
$p = 0.14$	$p = 6.79 \cdot 10^{-18}$	$p = 1.08 \cdot 10^{-9}$	$p = 1.34 \cdot 10^{-12}$
$A = 0.41$	$A = 1$	$A = 0.85$	$A = 0.91$

6 Discussion

There is one issue that deserves attention. We have observed that VE-GP reveals its potential when the aggregate functions involved in the target generate variable values among fitness cases. To clarify the statement, we have considered



(a) K5 training (b) K5 testing (c) B1 training

(d) B1 testing (e) B2 training (f) B2 testing

(g) B3 training (h) B3 testing

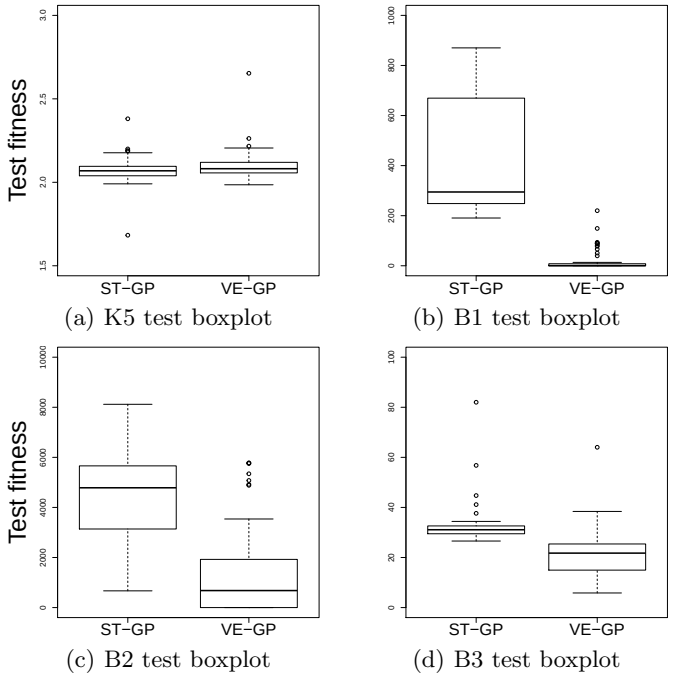
Fig. 2. ST-GP and VE-GP fitness evolution plots. The bars represent the first and the third quartile. K5 through a) and b), B1 through c) and d), B2 through e) and f) and B3 through g) and h) .

the following benchmark problem

$$BBN = \texttt{VSUMW}(-5.41, \texttt{VprW}(4.9, \texttt{VdivW}(\texttt{VSUMW}(\texttt{V\_max}(X4), X3), X1)))$$

where $X1$ and $X2$ are random numbers between 0 and 1, $X3$ is a vector of random numbers between -1 and 1 of length 10 and $X4$ is a vector of random numbers between 2 and 3 of length 10. In this case, the vectorial term $\texttt{V\_max}(X4)$ returns a value close to 3 for all the observations. Therefore ST-GP can approximate it with a constant despite the absence among primitives of the aggregate functions. We have investigated the performances of both GP method on this problem as before. The Wilcoxon rank test on the final fitness of the test set returns a p-value equal to 0.12 demonstrating that there is no difference in performances as expected.

The vector based approach of VE-GP suggests further developments and different applications from what we have treated in this paper. Although this



(a) K5 test boxplot (b) B1 test boxplot

(c) B2 test boxplot (d) B3 test boxplot

Fig. 3. ST-GP and VE-GP fitness boxplots for test set.

work is specially meant for time-series vectors, the technique is in fact usable for every dataset that involves variables whose suitable representation is a vector. Moreover, we can even think about a matricial approach of genetic programming. Some variables, in fact, may change in space and time. Therefore, a matrix is a suitable representation for its records. Additionally, it is possible to group vectorial variables to catch intra-records dependencies. As for the VE-GP, we will introduce new functions inspired by classical matrix operations such as matrix row by column multiplication or determinant and we will enhance the different blocks of GP structure in order to exploit this new representation.

7 Conclusion

The objective of this paper was to study the potential of a new approach of genetic programming to predict panel data. VE-GP is able to deal with different data structures, heterogeneous both in source and scale. Therefore it is possible to preserve the true nature of sequential variables, and it is no longer necessary to untie them in a classical panel dataset. The main contribution of this approach is however the capability to extract the most informative features of the behaviour of a time series variable during the evolution.

In order to characterize suitable problems for VE-GP we have chosen benchmark problems in which the algorithm would perform well. However, the idea of

VE-GP approach was suggested by panel datasets that represents many problems which stem from the real world. Therefore, to claim that VE-GP reveals advantages, we plan to apply it on already studied real problems so that it will be possible to compare the new results with the ones of previous works [12].

Future efforts should be made to develop a matricial approach of genetic programming in order to analyse how the algorithm behaves when posed with a different group of fitness cases belonging to the same datasets. By means of VE-GP we have investigated the first way of combining records and we would like to move further towards a single record containing all the observations as the training set.

Acknowledgments. This work was partially supported by FCT through funding of LASIGE Research Unit (UID/CEC/00408/2019), BioISI Research Unit (UID/MULTI/04046/2013), and projects INTERPHENO (PTDC/ASP-PLA/28-726/2017), PERSEIDS (PTDC/EMS-SIS/0642/2014), OPTOX (PTDC/CTA-AMB/30056/2017), BINDER (PTDC/CCI-INF/29168/2017), GADgET (DSAIPA/DS/0022/2018) and PREDICT (PTDC/CCI-CIF/29877/2017).

References

1. Poli, R., Langdon, W., McPhee, N.: A field guide to genetic programming. Lulu Enterprises, UK Ltd (2008). https://doi.org/10.1007/s10710-008-9073-y
2. Dermofal, D.: Time-series cross-sectional and panel data models. Spatial Analysis for the Social Sciences **32**, 141–157 (2015). https://doi.org/10.1017/CBO9781139051293.009
3. Guo, H., Jack, L.B., Nandi, A.K.: Automated feature extraction using genetic programming for bearing condition monitoring. In: Proceedings of the 14th IEEE Signal Processing Society Workshop Machine Learning for Signal Processing. pp. 519–528 (2004). https://doi.org/10.1109/MLSP.2004.1423015
4. De-Falco, I., Della-Cioppa, A., Tarantino, E.: A genetic programming system for time series prediction and its application to el niño forecast. Soft Computing: Methodologies and Applications **32**, 151–162 (2005). https://doi.org/10.1007/3-540-32400-3_12
5. Holladay, K., Robbins, K.A.: Evolution of signal processing algorithm using vector based genetic programming. In: 15th International Conference on Digital Signal Processing. pp. 503–506 (2007). https://doi.org/10.1109/ICDSP.2007.4288629
6. Bartashevich, P., Bakurov, I., Mostaghim, S., Vanneschi, L.: Evolving PSO algorithm design in vector fields using geometric semantic GP. In: GECCO 2018: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 262–263 (2018). https://doi.org/10.1145/3205651.3205760
7. Alfaro, E.C., Sharman, K., Esparcia-Alcázar, A.: Genetic programming and serial processing for time series classification. Evolutionary Computation **22**, 265–285 (2013). https://doi.org/10.1162/EVCO_a_00110
8. Silva, S., Almeida, J.: GPLAB a genetic programming toolbox for MATLAB (2007), http://gplab.sourceforge.net/index.html
9. McDermott, J., O'Reilly, U.M., Luke, S., White, D.: A community-led effort towards improving experimentation in genetic programming, http://gpbenchmarks.org/

10. Vargha, A., Delaney, H.D.: A critique and improvement of the CL common language effect size statistics of McGraw and Wong. Journal of Educational and Behavioral Statistics **25**(2), 101–132 (2000). https://doi.org/10.2307/1165329
11. Luke, S., Panait, L.: Lexicographic parsimony pressure. In: GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 829–836 (2002)
12. Bisanzio, D., Giacobini, M., Bertolotti, L., Mosca, A., Balbo, L., Kitron, U., Vasquez-Prokopec, G.M.: Spatio-temporal patterns of distribution of West Nile virus vectors in eastern Piedmont region, Italy. Parasites & Vectors **4:230** (2011). https://doi.org/10.1186/1756-3305-4-230