Evolutionary computation based nonlinear transformations to low dimensional spaces for sensor data fusion and visual data mining

Valdés, Julio J.

Publisher's version / Version de l'éditeur:
https://doi.org/10.1109/CEC.2010.5585951
2010 IEEE Congress on Evolutionary Computation (CEC), pp. 1-8, 2010-07-23

Evolutionary Computation Based Nonlinear Transformations to Low Dimensional Spaces for Sensor Data Fusion and Visual Data Mining

Julio J. Valdés, *Senior Member, IEEE*

***Abstract*— Data fusion approaches are nowadays needed and also a challenge in many areas, like sensor systems monitoring complex processes. This paper explores evolutionary computation approaches to sensor fusion based on unsupervised nonlinear transformations between the original sensor space (possibly highly-dimensional) and lower dimensional spaces. Domain-independent implicit and explicit transformations for Visual Data Mining using Differential Evolution and Genetic Programming aiming at preserving the similarity structure of the observed multivariate data are applied and compared with classical deterministic methods. These approaches are illustrated with a real world complex problem: Failure conditions in Auxiliary Power Units in aircrafts. The results indicate that the evolutionary approaches used were useful and effective at reducing dimensionality while preserving the similarity structure of the original data. Moreover the explicit models obtained with Genetic Programming simultaneously covered both feature selection and generation. The evolutionary techniques used compared very well with their classical counterparts, having additional advantages. The transformed spaces also help in visualizing and understanding the properties of the sensor data.**

I. Introduction

Data fusion in general and sensor data fusion in particular, are necessary. The progress in sensor and communication technologies demand generic and robust combination of monitored data coming from sensor nodes and networks [1]. It becomes increasingly important in large scale sensor system to make data transmission, storage and interpretation more efficient. In this sense dimensionality reduction [2] and relevant information finding while preserving as much internal properties of the data as possible, becomes a challenging but important goal. The reduction of the dimensionality of the data benefits the application of many processing techniques as it alleviate the *curse of dimensionality* and makes the application of the methods more efficient and effective. On the other hand, it makes possible the *visual interpretation* of the data and allows a more direct involving of domain experts and decision makers, in the spirit of Visual Data Mining and Exploratory Analysis. Another important benefits are the reduction of the noise level and redundancies in the data.

Two main approaches are feature selection and feature generation. Whereas in the former the objective is to find (smaller) subsets of the original features with as much descriptive or discriminatory power as possible, the later tries to produce new (fewer) features as functions of the original ones, hence creating a new space of dimension equal to the number of created features. Mathematically, feature selection works with subspaces of the original one, whereas feature generation works with new spaces created by transformations of the original. These mappings may be linear, like the classical principal components of factor analysis [3], or nonlinear, which are more flexible and powerful, but also more complex. A generic nonlinear approach to sensor data aggregation was introduced in [4] and applied to real-world complex sensor data from auxiliary aircraft engines. Using classical (deterministic) optimization methods it was shown that a substantial dimensionality reduction can be achieved.

This paper extends the study to the use of computational intelligence methods (in particular, evolutionary computation algorithms (EC)) for nonlinear transformations into low dimensional spaces with dimensionality reduction and Visual Data Mining purposes. Differential Evolution (DE) and Genetic Programming (GP) are used to obtain implicit and explicit mappings. The preliminary results show that dimensionality reduction can be achieved at levels totally equivalent to those obtained with classical deterministic methods. Also, feature selection is achieved when using explicit analytic models from which the relevant variables can be readily exposed and their importance quantified. The paper is organized as follows: Section II reviews the use of nonlinear transformations for data fusion with classical methods. Section III introduces two evolutionary approaches for the same purpose (DE and GP). Section IV briefly presents the case study (Auxiliary Power Units (engines) in aircrafts). Section V describes the experimental framework. Section VI presents the results and Section VII the conclusions.

II. Nonlinear Space Transformations for Information Fusion

The construction of a smaller feature space can be performed via a nonlinear transformation that maps the original set of N-dimensional objects under study $\mathcal{O}$ into another space $\hat{\mathcal{O}}$ of smaller dimension $\hat{d} < N$. An intuitive metric should be used for Visual Data Mining purposes, besides dimensionality reduction. A feature generation approach of this kind implies information losses and non-linear transformations ($\varphi : \mathcal{O} \rightarrow \hat{\mathcal{O}}$) are required. This approach has been used for data representation and Visual Data Mining (knowledge and data exploration) [5]. There are essentially

Julio J. Valdés is with the National Research Council Canada, Institute for Information Technology, 1200 Montreal Rd. Bldg M50, Ottawa, ON K1A 0R6, Canada (phone: 1-613-993-0887; fax: 1-613-993-0215; email: julio.valdes@nrc-cnrc.gc.ca).

three kinds of spaces generally sought [6]: *i)* spaces preserving the structure of the objects as determined by the original set of attributes or other properties (unsupervised approach), *ii*) spaces preserving the distribution of an existing class or partition defined over the set of objects (supervised approach), and *iii*) hybrid spaces. Data structure is one of the most important elements to consider and it can be approached by looking at *similarity relationships* [7], [8] between the objects, as given by the set of original attributes [5]. From this point of view, transformations φ can be constructed that minimize error measures of information loss [9], based on similarities or distances.

If $\delta(\vec{x},\vec{y})$ is a dissimilarity measure between any two objects $\vec{x}\vec{y} \in \mathcal{O}$, and $\zeta(\hat{\vec{x}},\hat{\vec{y}})$ is another dissimilarity measure defined on objects $\hat{\vec{x}},\hat{\vec{y}} \in \hat{\mathcal{O}}$ ($\hat{\vec{x}} = \varphi(\vec{x}), \hat{\vec{y}} = \varphi(\vec{y})$), a frequently used error measure associated to the mapping φ is the Sammon error [9]:

$$\mathcal{S}_e = \frac{1}{\sum\limits_{\vec{x}\neq\vec{y}} \delta(\vec{x},\vec{y})} \frac{\sum\limits_{\vec{x}\neq\vec{y}} \Big(\delta(\vec{x},\vec{y}) - \zeta(\hat{\vec{x}},\hat{\vec{y}})\Big)^2}{\delta(\vec{x},\vec{y})} \qquad (1)$$

A. Computation of the Nonlinear Space with Classical Methods

In order to optimize Eq. 1 the classical Fletcher-Reeves algorithm (FR) is used, which is a well known technique used in deterministic optimization [10]. It assumes that the function to optimize can be approximated as a quadratic form in the neighborhood of a N dimensional point $\mathbf{P}$ and exploits the information contained in the partial derivatives of the objective function. This kind of optimization is prone to local extrema entrapment, therefore it is recommended to try different random initial parameter vectors.

The φ mappings obtained using the previously described approach are *implicit*, as the images of the transformed objects are computed directly and the algorithm does not provide a function representation. The accuracy of the mapping depends on the final error obtained in the optimization process. Explicit mappings can however be obtained from these solutions using neural networks, genetic programming (Section III-A), and other techniques. In general φ is a nonlinear function and in order to compare results from transformations obtained with different algorithms or different initializations, a canonical representation is preferred. It can be obtained by performing a principal component transformation $\mathcal{P}$ after φ, so that the overall transformation is given by the composition

$$\hat{\varphi} = (\varphi \centerdot \mathcal{P}) \qquad (2)$$

referred to as the canonical mapping. Since $\mathcal{P}$ does not change the dimension of the new space, an advantage of the canonical mapping is to make easier the comparison of different solutions. It also contributes to the interpretability of the new variables, as they have a monotonic distribution of the variance. The images of the mapped objects can be used for the construction of a 3D model (e.g. using virtual reality), for visual data mining and data exploration (pattern detection, cluster assessment, etc), as in this paper.

III. Computation of the Nonlinear Space with Computational Intelligence Methods

This task can be performed also using evolutionary computation methods.

1) Differential Evolution: Differential Evolution [11], [12] is a kind of evolutionary algorithm working with real-valued vectors, and it is relatively less popular than genetic algorithms. However, it has proven to be very effective in the solution of complex optimization problems [13], [14]. Like other EC algorithms, it works with populations of individual vectors (real-valued), and evolves them. Many variants have been introduced, but the general scheme is as follows:

step 0 Initialization: Create a population $\mathcal{P}$ of random vectors in $\Re^n$, and decide upon an objective function $f : \Re^n \rightarrow \Re$ and a strategy $\mathcal{S}$, involving vector differentials.
step 1 Choose a target vector from the population $\vec{x}_t \in \mathcal{P}$.
step 2 Randomly choose a set of other population vectors $\mathcal{V} = \{\vec{x}_1, \vec{x}_2, \ldots\}$ with a cardinality determined by strategy $\mathcal{S}$.
step 3 Apply strategy $\mathcal{S}$ to the set of vectors $\mathcal{V} \cup \{\vec{x}_t\}$ yielding a new vector $\vec{x}_{t'}$.
step 4 Add $\vec{x_t}$ or $\vec{x_{t'}}$ to the new population according to the value of the objective function f and the type of problem (minimization or maximization).
step 5 Repeat steps 1-4 to form a new population until termination conditions are satisfied.

There are several variants of DE which can be classified using the notation $DE/x/y/z$, where x specifies the vector to be mutated, y is the number of vectors used to compute the new one and z denotes the crossover scheme. Let F be a scaling factor, $\mathcal{C}_r \in \Re$ be a crossover rate, D be the dimension of the vectors, $\mathcal{P}$ be the current population, $N_p = card(\mathcal{P})$ be the population size, $\vec{v}_i$, $i \in [1, N_p]$ be the vectors of $\mathcal{P}$, $\vec{b}_{\mathcal{P}} \in \mathcal{P}$ be the population's best vector w.r.t. the objective function f and $r, r_0, r_1, r_2, r_3, r_4, r_5$ be random numbers in $(0, 1)$ obtained with a uniform random generator function $rnd()$ (the vector elements are $\vec{v}_{ij}$, where $j \in [0, D)$). Then the transformation of each vector $\vec{v}_i \in \mathcal{P}$ is performed by the following steps:

step 1 Initialization: $j = (r \cdot D)$, $L = 0$
step 2 $while(L < D)$
step 3 $if((rnd() < \mathcal{C}_r)||L == (D-1))$
/* create a new trial vector. For example, as: */
$\vec{t}_{ij} = \vec{b}_{\mathcal{P}j} + F \cdot (\vec{v}_{r_1 j} + \vec{v}_{r_2 j} - \vec{v}_{r_3 j} - \vec{v}_{r_4 j})$
step 4 $j = (j+1) \bmod D$
step 5 $L = L + 1$
step 6 goto 2
step 7 stop

Many particular strategies have been proposed and they differ in the way the trial vector is constructed (step 3 above). The

ones used in this paper are (see Table. I):

$$\begin{array}{ll} DE/ & best/2/bin \\ t_{ij}^{\mathcal{P}} & = v_{bj}^{\mathcal{P}} + F \times (v_{r_1 j}^{\mathcal{P}} + v_{r_2 j}^{\mathcal{P}} - v_{r_3 j}^{\mathcal{P}} - v_{r_4 j}^{\mathcal{P}}) \\ DE/ & rand/2/bin \\ t_{ij}^{\mathcal{P}} & = v_{r_5 j}^{\mathcal{P}} + F \times (v_{r_1 j}^{\mathcal{P}} + v_{r_2 j}^{\mathcal{P}} - v_{r_3 j}^{\mathcal{P}} - v_{r_4 j}^{\mathcal{P}}) \\ DE/ & rand/1/exp \\ t_{ij}^{\mathcal{P}} & = v_{r_3 j}^{\mathcal{P}} + F \times (v_{r_1 j}^{\mathcal{P}} - v_{r_2 j}^{\mathcal{P}}) \\ DE/ & best/1/exp \\ t_{ij}^{\mathcal{P}} & = v_{bj}^{\mathcal{P}} + F \times (v_{r_1 j}^{\mathcal{P}} - v_{r_2 j}^{\mathcal{P}}) \\ DE/ & best/2/exp \\ t_{ij}^{\mathcal{P}} & = v_{bj}^{\mathcal{P}} + F \times (v_{r_1 j}^{\mathcal{P}} + v_{r_2 j}^{\mathcal{P}} - v_{r_3 j}^{\mathcal{P}} - v_{r_4 j}^{\mathcal{P}}) \\ DE/ & rand - to - best/1/exp \\ t_{ij}^{\mathcal{P}} & = v_{ij}^{\mathcal{P}} + F \times (v_{bj}^{\mathcal{P}} + v_{ij}^{\mathcal{P}} - v_{r_1 j}^{\mathcal{P}} - v_{r_2 j}^{\mathcal{P}}) \end{array} \quad (3)$$

where $t_{ij}^{\mathcal{P}}$ is a new trial vector and b is the index to the best vector $\vec{b}_{\mathcal{P}}$. Some of them, like the $DE/best/2/bin$ have been reported as producing good results in a wide variety of test problems [14].

When using DE for solving the nonlinear mapping problem described by Eq.1 the dimension of the vectors in the DE algorithm was set to $D = card(\mathcal{O}) \times \hat{d}$ in order to construct a representation in which each DE vector provides a candidate solution to Eq.1, which clearly be an implicit mapping.

A. Genetic Programming

Genetic programming (GP) techniques aim at evolving computer programs. They are an extension of the Genetic Algorithm introduced in [15] and further elaborated in [16], [17] and [18]. The algorithm starts with a set of randomly created computer programs. This initial population goes through a domain-independent breeding process over a series of generations. Genetic programming combines the expressive high level symbolic representations of computer programs with the search efficiency of the genetic algorithm. Those programs which represent functions are of particular interest and can be modeled as $y = F(x_1, \cdots, x_n)$, where $(x_1, \cdots, x_n)$ is the set of independent or predictor variables, and y the dependent or predicted variable, so that $x_1, \cdots, x_n, y \in \mathbb{R}$, where $\mathbb{R}$ are the reals. The function F is built by assembling functional subtrees using a set of predefined primitive functions (the Function Set), defined beforehand. In general terms, the model describing the program is given by $y = F(\vec{x})$, where $y \in \mathbb{R}$ and $\vec{x} \in \mathbb{R}^n$. Most implementations of genetic programming for modeling fall within this paradigm but for some problems vector functions are required. A GP based approach for finding vector functions was presented in [19]. In these cases the model associated to the evolved programs is $\vec{y} = F(\vec{x})$, which allows for the simultaneous estimation of several dependent variables $\vec{y}$ from a set of independent variables $\vec{x}$. Note that these are *not* multi-objective problems, but problems where the fitness function depends on vector variables. The mapping problem between vectors of two spaces of different dimension (n and m) is one of that kind. In this case a transformation like $\psi : \mathbb{R}^n \to \mathbb{R}^m$, mapping vectors $\vec{x} \in \mathbb{R}^n$ to *vectors* $\vec{y} \in \mathbb{R}^m$ would allow a reformulation of Eq. 1:

$$\mathcal{S}_e = \frac{1}{\sum_{i<j} \delta_{ij}} \frac{\sum_{i<j} (\delta_{ij} - d(\vec{y}_i, \vec{y}_j))^2}{\delta_{ij}}, \quad (4)$$

where $\vec{y}_i = \psi(\vec{x}_i)$, $\vec{y}_j = \psi(\vec{x}_j)$.

The evolution has to consider populations of *forests* such that the evaluation of the fitness function depends on the set of trees within a forest [19]. In these cases, the cardinality of any forest within the population is equal to the dimension of the target space m.

Gene Expression Programming (GEP) [20], [21] is one of the many variants of GP and has a simple string representation. In the GEP algorithm, the individuals are encoded as simple strings of fixed length with a head and a tail, referred to as chromosomes. Each chromosome can be composed of one or more genes which hold individual mathematical expressions that are linked together to form a larger expression.

For the research described in this paper, the extension of the GEP algorithm which supports vector functions was used [19]. The GEP implementation is an extension to the ECJ System [22].

When genetic programming is used for solving Eq. 4 by estimating ψ explicitly, the terminal set for the GP algorithm may become large (potentially very large) if the dimension of the original space (the number of attributes of the data vectors) is large and the function set is rich. This poses a big challenge to the GP search process, as the search space is huge (perhaps infinite) whereas the number of functions generated during the evolutionary process is necessarily is a tiny fraction of it. However, there are important advantages of GP solutions to Eq. 4 if they provide a reasonable low error level. The explicit nature of the solution makes it a *transparent* model rather than a black-box one, which is the case of neural networks, svm's and other computational intelligence methods. On the other hand, the union of the set of arguments of the vector equations associated to the mapping ψ provides a subset of the original variables found as relevant, since they are the chosen ones for building the GP model equations. Of no smaller advantage is the fact that by retaining not a single, but a set of k-best GP solutions, an ensemble model (committee of experts) can be built. In the same way, the solutions can be boosted or jointly used with a variety of schemes. Yet another potential advantage is a quantitative assessment of the importance of the selected variables, which is given by the elements of the Jacobian matrix as a sensitivity measure.

$$J_\psi(x_1, \cdots, x_N) = \frac{\partial(y_1, \cdots, y_{\hat{d}})}{\partial(x_1, \cdots, x_N)} \quad (5)$$

IV. A Case Study: Auxiliary Power Unit Sensor Data Fusion

The Auxiliary Power Unit(APU) engines on commercial aircraft provide electrical power and air conditioning in the cabin prior to the starting of the main engines and also supply the compressed air required to start the main engines when the aircraft is ready to leave the gate. APUs are highly reliable but they occasionally fail to start due to failures of the

APU starter motor. When this happens, additional equipment such as generators and compressors must used to replace APU's functionalities, which implies significant costs, delays or flight cancellation. Accordingly, airlines are very much interested in monitoring the health of the APU starter motors and proceed with preventive maintenance whenever a failure is suspected. The data comes from sensors installed at strategic locations in the APU which monitors various phases of operation. The ultimate objective is to develop predictive models that can accurately determine the remaining useful life of the starter motors. A generic methodology to tackle this problem was proposed by [23] but non-linear data fusion could help in further improve the results obtained and also facilitate the visualization and interpretation of the data.

The dataset comes from APU starting reports containing 18 original attributes (5 symbolic, 11 numeric, and 2 for date and time of the event) collected around each occurrence of component failures. The analysis is based on data generated between a certain starting date prior to the failure. Statistical, time-series, and signal processing filters on the initial 11 numerical sensor measurements produced an additional collection of 188 new variables (features) for a total of 198 numerical variables (the dimension of the original space) and 238 time-stamp instances. With such a highly dimensional space, it is impossible to visualize the characteristics of the APU sensor data.

V. Experimental Settings

The data used come from a fleet of 35 commercial Airbus 320 series aircraft collected over a period of 10 years. ACARS (Aircraft Communications Addressing and Reporting System) Auxiliary Power Unit starting reports were made available. Among them, an example of an APU was used for experimentation. Eq. 2 was used for computing new 3-D spaces by nonlinear transformation of the original 198-D space of the observed and derived sensor data (via Eq. 1), for 238 time observations. The dissimilarity measure used has the form

$$\delta(\vec{x},\vec{y}) = \frac{1}{S(\vec{x},\vec{y})} - 1 \tag{6}$$

where $\vec{x},\vec{y}$ are any two vectors in the original p-dimensional space of sensor information and $S(\vec{x},\vec{y})$ is Gower's similarity coefficient [24] which for numeric variables is

$$S(\vec{x},\vec{y}) = \sum_{i=1}^{p} 1 - \frac{|x_i - y_i|}{Range(i)} \tag{7}$$

where x_i, y_i are the corresponding i-th vector components. A priori, there is no univocal indication about how many new dimensions are required in order to achieve a good similarity preservation between the original sensor space and the reduced target space for an arbitrary dataset. That is, how many dimensions would ensure an acceptable low mapping error. In the present case three dimensional target spaces were tried, as they would allow a visual inspection of the transformed sensor data obtained by different algorithms. In the transformed spaces, the Euclidean distance was used as the dissimilarity measure. These general settings were the ones used in [4] and were kept the same across the mapping methods used in this paper. The following subsections cover those settings specific to the individual algorithms

A. Classical Optimization Settings

For the Fletcher-Reeves method, the settings were those described in [4]: 10 initial random configurations in the target space were tried when computing $\hat{\varphi}$. The best mapping error solution was kept (in order to cope with local minima).

B. Differential Evolution Settings

Table I shows the experimental settings used for different runs of the DE algorithm. For every DE run, the 10 best chromosomes were retrieved. Population size was fixed to 100 vectors of length 714 (238 objects whose image in a 3-D space require 3 coordinates). The maximum number of generations was set to 15000 as well as a Sammon error threshold of 0.014. The idea was to asses the DE capabilities to approximate error levels of those obtained with classical methods like FR for the same data.

TABLE I
Experimental settings for the Differential Evolution algorithm (DE-FR). 3D and 1D spaces were computed

Parameter	Values
F	$\{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$
Cross-over Rate	$\{0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$
Population Size	$\{100\}$
Strategies	$DE/best/2/bin$, $DE/rand/2/bin$, $DE/rand/1/exp$, $DE/best/1/exp$, $DE/best/2/exp$, $DE/rand-to-best/1/exp$
Number of Trials	5

A total of 1470 runs covered the experimental settings discussed above, producing a total of 14700 candidate new spaces, as for each run the 10 best vectors were retained. From these first round of results, some few were selected for further exploration (see Section VI-B).

C. Genetic Programming Settings

The ECJ-GEP genetic programming experiments were performed using a population sizes of 1000 individuals. 100 different random seeds were used for each experiment and the best individual found was kept for each run. The total number of GP experiments was 500. The Function Set was composed by a *Basic Set* composed of arithmetic functions: $\{+, -, *, /\}$ with weights given by $\{2, 1, 1, 1\}$ and additional functions like the power ($x \wedge y = x^y$) and the Logistic ($\mathcal{L}(x) = 1/(1+e^{-x})$) were used in some cases as shown in Table. II.

The remaining algorithm parameters were fixed at the following suggested values [21]: genes/chromosome = 5, gene headsize = 5, elitism = 3 individuals, constants = allowed (in $[-1, 1]$), probabilities: inversion = 0.1, mutation = 0.044, istransposition = 0.1, ristransposition-prob = 0.1, onepointrecomb-prob = 0.3, twopointrecomb-prob = 0.3,

Parameter	exp-2	exp-3	exp-4	exp-5	exp-6
nbrGenerations	10000	10000	20000	100000	20000
nbr of Genes	4	4	4	4	6
Head Size	8	8	8	8	16
Additional Functions		$\wedge$	$\wedge, \mathcal{L}$	$\wedge, \mathcal{L}$	$\wedge, \mathcal{L}$
Weights		1	1, 1	1, 1	1, 1

TABLE II

EXPERIMENTAL SETTINGS FOR DIFFERENT GENETIC PROGRAMMING EXPERIMENTS (500 IN TOTAL).

generecomb-prob = 0.1, genetransposition-prob = 0.1, rnc-mutation= 0.01, dc-mutation-prob = 0.044, dc-inversion= 0.1, dc-istransposition = 0.1.

VI. RESULTS

A. Mapping with classical deterministic methods

The implicit canonical 3-D mapping of the 198-D vectors (238 in total), corresponding to the APU-1 engine data obtained in [4] is shown in Fig. 1(Top). It is impossible to represent a 3D model on hard media. Therefore fixed snapshots of the actual 3D volume are displayed on the figure. When comparing them it should be remembered that the orientation of the coordinate axis is irrelevant as distance is invariant to rotation.

The small mapping error obtained (0.01315), indicates that despite of the large amount of sensor space compression due to nonlinear dimensionality reduction, the amount of information lost is small. Therefore, the new 3-D space provides a reasonable representation of the overall data structure from the point of view of the preservation of the similarity relationships defined by the original sensor variables.

For comparison purposes, the vectors have been classified into three categories (1: timeToFailure $\in (-\infty, -100]$ days, 2: timeToFailure $\in [-99, -34]$ days, 3: timeToFailure $\in [-33, 0]$ days), as was done in [4], where transparent closed surfaces wrapping the vectors corresponding to the given classes were added (Fig. 1). It is important to recall that the computed mappings are unsupervised, therefore this time-to-failure based class information was not used in the process by any of the methods considered. However, the similarity preservation space recognizes well the time evolution of the APU engine status from a far-from-failure states to close-to-failure situations.

From the point of view of the time evolution, a more detailed view is provided by a 3-D polyline joining consecutive observations starting from the initial vector (237 days before failure), to the final one at failure time. This polyline represents the trajectory of the 198 individual time series from the original sensor space and enables a better understanding of the system's evolution as failure contributing factors cumulate and grow in influence on the process. The X-axis in the canonical nonlinear 3-D space (the one with the largest variance), clearly shows the time evolution of the failure classes: timeToFailure $\in (-\infty, -100]$ days is located at the low values end, whereas the timeToFailure $\in [-99, -34]$ days and timeToFailure $\in [-33, 0]$ classes follows as the X-values increase. Therefore, the nonlinear X-axis represents an 'aging factor' of the system. The same convention was used for representing the DE and GP results.

B. Differential Evolution Results

The overall distribution of the mapping error for the 14700 DE-computed new spaces, is shown in Fig. 2. The distribution is bimodal and highly skewed towards low error

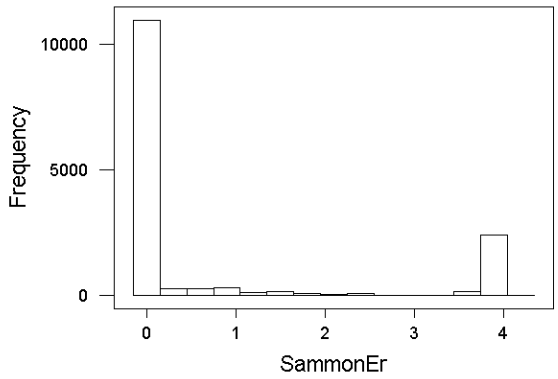

Fig. 2. Mapping error distribution for the spaces obtained with Differential Evolution in $15,000$ or less generations.

values, clearly indicating that the overwhelming majority of the spaces obtained are of good quality and that the DE algorithm is robust. The error range is $[0.0140, 4.0546]$ with a median of 0.0264, which is in the order of the best solution obtained with the classical FR algorithm (0.01314). Considering the high dimensionality of the DE vectors (714) this behavior is remarkable. In particular, 267 solutions achieve the error level threshold of 0.014 in $15,000$ or less generations. For these, the distribution of the actual number of generations needed is shown in Fig. 3, which

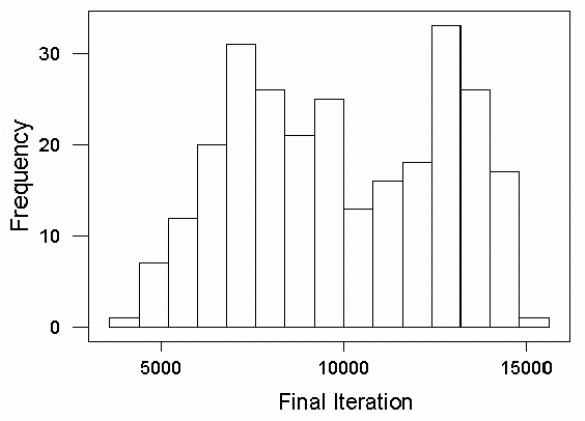

Fig. 3. Distribution of the number of generations required for the spaces obtained with Differential Evolution to achieve an error threshold of 0.014 in $15,000$ or less generations.

has a bimodal character with modes located around $7,500$ and $13,000$ generations. Most of the runs leading to low mapping error models required less than $10,000$ generations.

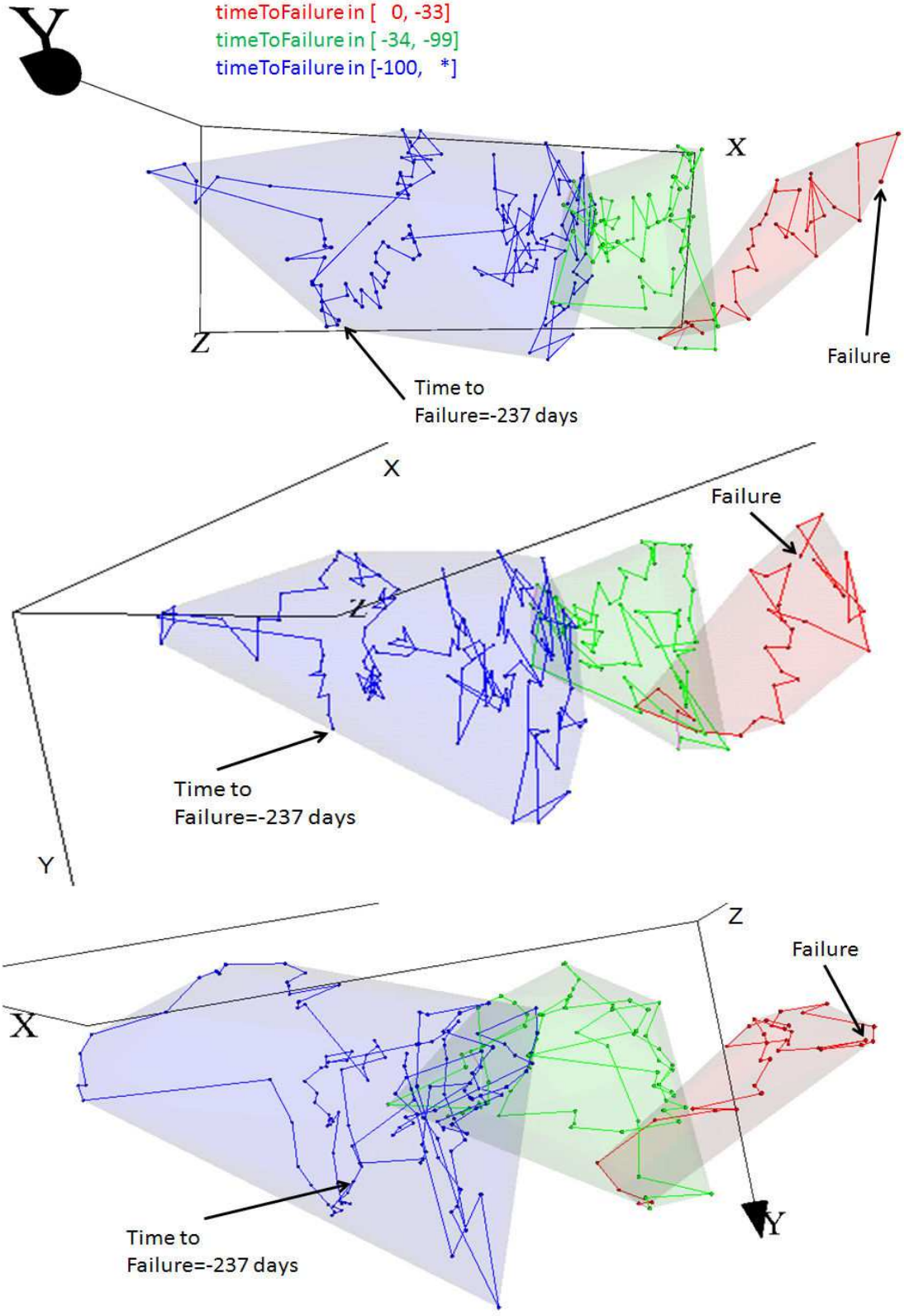

Fig. 1. $\mathbb{R}^{198} \rightarrow \mathbb{R}^3$ mapping of the APU data. A 3D polyline links consecutive points along the original 198-dimensional time series from the starting point (Time to Failure = -237) to the last (Time to Failure = 0). Semi-transparent surfaces wrap the three main classes: Time to Failure within 33 days or less, within 33 and 100 days and more than 237 days. The classes are reasonably well distinguished in the 3D space resulting from the nonlinear information fusion process. Top: Space obtained with the classical Fletcher-Reeves method (Error = 0.01315). Middle: Space obtained with Differential Evolution (Error = 0.01315). Bottom: Space obtained with Genetic Programming (Error = 0.0338).

Among these, some converged rapidly to the error threshold in less than $5,000$ generations. In particular, one achieved the error threshold in 4163 generations and was allowed to continue evolving. At 10336 generations it achieved an error of 0.01315, which is the same as the one obtained with the FR method. Considering the high dimensionality of the DE vector space, the fact that many solutions come close to the best result obtained with a classic technique which exploits the knowledge of the partial derivatives of the objective function (which DE does not) is remarkable. The resulting 3D space corresponding to that model is shown in Fig. 1(Middle). The effect of various DE parameters considered on the convergence towards accurate mappings is shown in Table III.

DE Strategy	Freq	F	Freq	Crossover Rate	Freq
DE/best/1/exp	85	0.4	67	0.9	83
DE/rand-to-best/1/exp	63	0.5	61	0.8	55
DE/best/2/exp	44	0.3	57	0.6	44
DE/best/2/bin	28	0.2	43	0.7	38
DE/rand/1/exp	25	0.6	28	0.5	27
DE/rand/2/bin	22	0.7	11	0.3	11
————-	-	-	-	0.4	9

TABLE III

DISTRIBUTION OF THE DE CONTROLLING PARAMETERS FOR THE MODELS WITH MAPPING ERROR UNDER THE 0.014 THRESHOLD.

The $DE/best/1/exp$ and the $DE/rand{-}to{-}best/1/exp$ strategies were clearly more prone to drive the evolutionary process towards regions of the search space were high accurate mappings are found. F factors are in the $[0.3, 0.5]$ range and Crossover ratios larger than 0.5 lead to good results. Main features of the data distribution in the new space, like the relative ordering and trend of the broad classes defining the system's state according to the Time to Failure (a variable not considered in the computation of the mapping), are clearly expressed in Fig. 1(Middle) as well as the location of the end states (from -237 days to the failure point). These results show that DE is capable of performing at a level comparable to classical deterministic optimization in providing suitable low dimension implicit mappings.

C. Genetic Programming Results

The general behavior of the mapping error for all GP experiments is shown in Fig. 4. The distribution is skewed towards the low error values which is a good behavior for the algorithm. The error range is $[0.03382, 0.09027]$ with mean and median of 0.05532 and 0.05450 respectively. This error range is considerably smaller than the one obtained with DE after almost three times more runs. However, DE was capable to achieve a minimum equal to the one obtained with the FR method, whereas the best GP result was 0.0338 (within experiment 5), which is two times larger. The breakdown of the error values for the individual experiments is shown in the boxplots of Fig. 5. There is not much variation between the results produced by experiments 2 and 3 which

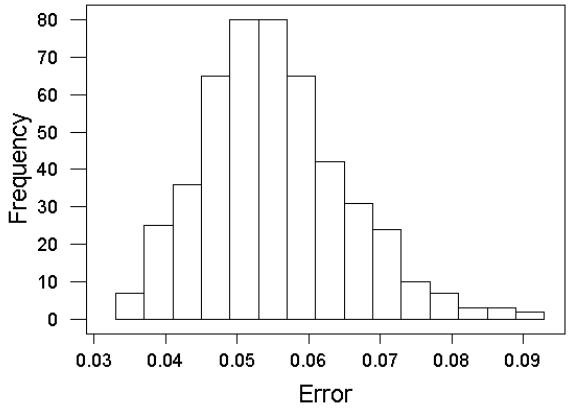

Fig. 4. Error distribution for the Genetic Programming experiments.

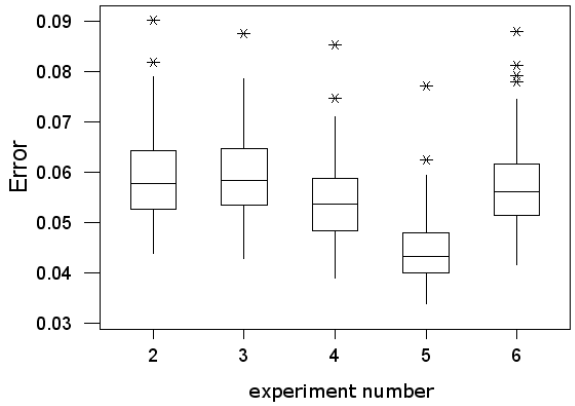

Fig. 5. Mapping error distributions for the GP experiments (See Table. II).

differ in the number of genes (allowing larger mathematical expressions) and in the introduction of the power function. However, performance improves by increasing the number of generations (more search) and the addition of the logistic function as shown by experiments 4 and 5. Experiment 6, was conceived as a derivation of 4 in the sense of providing room for forming longer and more complex expressions (increasing the number of genes and the chromosome head size). The performance improvement of experiment 5 with respect to 6 suggests that not too large or complex expressions are necessary beneficial. The results of experiment 5 seems to indicate that the amount of search made is the crucial factor (exps 4 and 5 differ only in the number of generations). Not only its mean, median, minimum and maximum are shifted towards lower error values, but also the interquartile range is smaller than in all other experiments. The best GP model representing the explicit mapping ψ is given by Eq. 8. When used in Eq. 4 the resulting 3D space is shown in Fig. 1(Bottom) with a mapping error of 0.0338. This error, although more than two times larger than those of FR and DE is still small enough as to represent a small information loss. The main characteristics of the associated 3D space in terms of the distribution of the time-to-failure classes and the location of the start and end states of the process

(Time to Failure -237 days and Time to failure 0), are the same as what is obtained with the other two methods. However with GP there is the additional advantage of having an analytical model (white box) in contradistinction with the implicit mappings, black-box models given by FR and DE.

$$\begin{aligned}
\psi_x =&(((v_{186}/v_{170}) + (v_{196}/v_{170})) \\
&+(((v_{75} - (v_{80} - v_{170})) \\
&+((v_{90}/v_{155}) + v_{191}))/v_{90})) \\
&+((v_{90} + (v_{10} + (v_{175} + v_{165})))/v_{71}) \\
\psi_y =&(((\mathcal{L}((v_{85} + v_{108})/(v_{26} + v_{165})) + v_{38}) \\
&+\mathcal{L}(\mathcal{L}((6.811566/v_{55}) - (v_{181}/v_{117})))) + \mathcal{L}(v_{52})) \\
&+\mathcal{L}(v_{118}) \\
\psi_z =&((\mathcal{L}(v_{14}/v_{94})/v_{159})^{\mathcal{L}(\mathcal{L}(v_{33}))} + (1/\mathcal{L}(v_{145}/v_{137})) \\
&+\mathcal{L}(v_{125}/(((v_{148} - v_{78}) + (v_{184} * v_{173})) \\
&+(v_{45} * v_{197}))) + \mathcal{L}(\mathcal{L}(v_{103}))
\end{aligned} \tag{8}$$

With new data, their images in the new space can be directly obtained by applying Eq. 8. Since the mappings obtained with FR and DE are implicit, it is necessary to merge the new data with the previous and recompute Eq. 1 every time. Another important side result of the GP approach is the feature selection that occurs as part of the evolutionary process. From the original 198 attributes defining the original space, only 35 appear as arguments of the ψ functions, meaning that the main similarity structure of the data can be represented with only 17.7% of the original variables, which is also an important dimensionality reduction per se. In terms of search effort, Eq. 8 was found at generation $99,409$. However, another solution obtained with the settings of experiment 4 achieved a mapping error of 0.03890 after 19225 generations, involving 42 variables. This is a fairly equivalent result but requiring only 19% of the evolutionary effort. GP can perform comparable to classical deterministic optimization in providing low dimension *explicit* mappings. The advantage is that both dimensionality reduction and feature selection can be made (not possible with FR or DE).

VII. Conclusions

Nonlinear unsupervised transformation approaches to data fusion were applied to a case study from the aerospace domain. Experiments for dimensionality reduction were conducted, indicating that nonlinear transformations are effective to reduce dimensionality while preserving the structure of the original data and relevant information. The results also demonstrated that the transformed spaces help in understanding the characteristics of the sensor data. EC techniques proved to compare well with classical methods, with the extra advantage of providing white box models which allows simultaneous feature selection and generation. However, EC solutions are more expensive that their classical counterparts. The results suggest that relationships could be established between the nonlinear variables of the transformed spaces with the time to failure of the APU unit, which will be further investigated.

References

[1] H. Durrant-Whyte, "Data fusion in sensor networks," in *Proc. IEEE International Conference on Video and Signal Based Surveillance AVSS '06*, 2006, pp. 39–39.

[2] I. Fodor, "A survey of dimension reduction techniques," Lawrence Livermore National Lab., CA (US), Tech. Rep., 2002.

[3] H. H. Harman, *Modern Factor Analysis*. University Of Chicago Press; 3 edition, 1976.

[4] J. Valdés, S. Letourneau, and C. Yang., "Data fusion via nonlinear space transformations," in *Proceedings of the First International Conference on Sensor Networks and Applications (SNA-2009)*. San Francisco, USA.: International Society for Computers and Their Applications (ISCA), November 4-6 2009.

[5] J. Valdés, "Virtual reality representation of information systems and decision rules: an exploratory technique for understanding data knowledge structure," in *Lecture Notes in Artificial Intelligence, LNAI*. Springer-Verlag, 2003, vol. 2639, pp. 615–618.

[6] J. Valdés and A. Barton, "Virtual reality spaces for visual data mining with multiobjective evolutionary optimization: Implicit and explicit function representationsmixing unsupervised and supervised properties," in *Proceedings of the IEEE Congress of Evolutionary Computation*, 2006, pp. 592–598.

[7] L. Chandon and S. Pinson, *Analyse typologique. Théorie et applications*. Masson, 1981.

[8] I. Borg, *Multidimensional similarity structure analysis*. New York, NY, USA: Springer-Verlag New York, Inc., 1987.

[9] J. W. Sammon, "A nonlinear mapping for data structure analysis," *IEEE Trans. Comput.*, vol. 18, no. 5, pp. 401–409, 1969.

[10] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C*, 2nd ed. Cambridge, UK: Cambridge University Press, 1992.

[11] R. Storn and K. Price, "Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces." ICSI, Tech. Rep. tr-09012, 1995.

[12] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution. A Practical Approach to Global Optimization.*, ser. Natural Computing Series. Springer Verlag, 2005.

[13] S. Kukkonen and J. Lampinen, "An empirical study of control parameters for generalized differential evolution." Kanpur Genetic Algorithms Laboratory (KanGAL), Tech. Rep. 2005014, 2005.

[14] R. Gämperle, S. D. Müller, and P. Koumoutsakos, "A parameter study for differential evolution." [Online]. Available: citeseer.ist.psu.edu/526865.html

[15] J. Koza, "Hierarchical genetic algorithms operating on populations of computer programs." in *Proceedings of the 11th International Joint Conference on Artificial Intelligence. San Mateo, CA*. Morgan Kaufmann, 1989, pp. 768–774.

[16] ——, *Genetic programming: On the programming of computers by means of natural selection.*, MIT Press, 1992.

[17] ——, *Genetic programming ii: Automatic discovery of reusable programs.*, MIT Press, 1994.

[18] J. Koza, D. Andre, and M. Keane, *Genetic programming ii: Automatic discovery of reusable programs.*, Morgan Kaufmann, 1999.

[19] J. Valdés, A. Barton, and R. Orchard, "Exploring medical data using visual spaces with genetic programming and implicit functional mappings," in *Proceedings of the Genetic and Evolutionary Computation. GECCO 2007 Conference*, London, July 7-11 2007.

[20] C. Ferreira, "Gene expression programming: A new adaptive algorithm for problem solving," *Journal of Complex Systems*, vol. 13, no. 2, pp. 87–129, 2001.

[21] ——, *Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence*, Springer Verlag, 2006.

[22] S. Luke, L. Panait, G. Balan, S. Paus, Z. Skolicki, E. Popovici, J. Harrison, J. Bassett, R. Hubley, , and A. Chircop, *ECJ, A Java-based Evolution Computing Research System*, Evolutionary Computation Laboratory, George Mason University., March 2007. [Online]. Available: http://www.cs.gmu.edu/~eclab/projects/ecj/

[23] S. Létourneau, C. Yang, and Z. Liu, "Improving preciseness of time to failure predictions: Application to APU starter," in *Proceedings of the 1st International Conference on Prognostics and Health Management*, Denver, Colorado, USA, October 2008.

[24] J. C. Gower, "A general coefficient of similarity and some of its properties," *Biometrics*, vol. 1, no. 27, pp. 857–871, 1973.