Approximate *k*-NN Delta Test minimization method using genetic algorithms: Application to time series[☆]

Fernando Mateo[*,a], Dušan Sovilj[b], Rafael Gadea[a]

[a] *Institute of Applications of Information Technologies and Advanced Communications, Universidad Politécnica de Valencia, Valencia, Spain*
[b] *Laboratory of Information and Computer Science, Helsinki University of Technology, Espoo, Finland*

Abstract

In many real world problems, the existence of irrelevant input variables (features) hinders the predictive quality of the models used to estimate the output variables. In particular, time series prediction often involves building large regressors of artificial variables that can contain irrelevant or misleading information. Many techniques have arisen to confront the problem of accurate variable selection, including both local and global search strategies. This paper presents a method based on genetic algorithms that intends to find a global optimum set of input variables that minimize the Delta Test criterion. The execution speed has been enhanced by substituting the exact nearest neighbor computation by its approximate version. The problems of scaling and projection of variables have been addressed. The developed method works in conjunction with MATLAB's Genetic Algorithm and Direct Search Toolbox. The goodness of the proposed methodology has been evaluated on several popular time series examples, and also generalized to other non-time-series datasets.

Key words: Genetic algorithm, Delta test, Variable selection, Approximate *k*-nearest neighbors, Variable scaling, Variable projection, Time series

1. Introduction

In many fields like science, industry and finance it is necessary to accurately predict future values of a time series. Some examples of problems that would benefit from an accurate prediction are: industrial processes, that can be modeled, predicted and controlled based on sensory data; natural phenomena, such as daily rainfall or seismic events; medical applications like the modeling of biological signals such as EEG or ECG; and financial problems like the prediction of stock market prices.

The correct estimation of future values of time series is usually affected by complex processes like random fluctuations, sudden trend changes, volatility and noise. The horizon of prediction and the number of available samples to obtain one or more future estimations are important issues too. When building a regressor, which can be understood as the number of past events used to predict their next one, the number of inputs to the model (which is translated as the size of the regressor) can become very large, depending on the periodicity of the particular time series. With large regressors, the learning procedure of the involved predictive models becomes slow and tedious.

Historically, the different models employed to estimate time series have been differentiated in two groups: linear and nonlinear methods. The most popular linear methods are based on the Box-Jenkins methodology [1]. They include autoregressive (AR) models, integrated (I) models, and moving average (MA) models. Their combination has given rise to autoregressive moving average (ARMA) models, autoregressive integrated moving average (ARIMA) models and their seasonal generalization (SARIMA) [2]. However, these models are too limited and simplistic for the average complexity of a time series. In contrast, nonlinear methods are more suitable for complex series that contain irregularities and noise, such as chaotic time series. There is abundant literature on nonlinear models for time series forecasting [3, 4, 5, 6, 7, 8]. Among the existing methods are neural networks [9, 10, 11, 12, 13, 14, 15], radial basis function networks [11, 16, 17, 18], support vector machines [19, 20, 21, 22], self organizing maps [23, 24] and other variants of these models [11, 25, 26, 27, 28]. However, building these models takes considerable computational time compared to linear models. Recently, several hybrid methods (ARIMA + fuzzy or neural networks) have been employed in the literature [29, 30, 31].

Both linear, nonlinear, and hybrid methods have the same purpose: to gather enough information from past samples to give a reliable prediction of the immediate future samples (short-term prediction) or give estimations about far-future samples (long-term prediction). Long term prediction (i.e. predicting multiple steps ahead towards the future) is usually more challenging because the accumulation of errors and inherent uncertainties of a multiple-step-ahead in time yields deteriorated estimates of future samples.

Time series prediction can be considered a modeling problem [32]. The inputs to the model are composed of a set of consec-

[☆] This work was supported by Spanish Ministry of Science and Innovation (MICINN) projects AGL 2004-07549-C05-02/ALI and FPA 2007-65013-C02-02, and a grant with reference BES-2005-9703.

[*] Corresponding author.
Email addresses: fermaji@upvnet.upv.es (Fernando Mateo), dusans@cis.hut.fi (Dušan Sovilj), rgadea@eln.upv.es (Rafael Gadea)

utive regressor instances, while the output is the next value or values of the series that have to be predicted after each regressor instance.

Normally, the size of the regressor is chosen according to the periodicity components of the time series. Consequently, time series with long periodic components may yield very large regressors that can be troublesome to handle by predictive models. Most modeling techniques do not deal well with datasets having a high number of input variables, due to the so called *curse of dimensionality* [33]. As the number of dimensions grows, the number of input values required to sample the solution space increases exponentially. Many real life problems present this drawback since they have a considerable amount of variables to be selected in comparison to the small number of observations. Therefore, efficient variable selection procedures are required to reduce the complexity while also improving the interpretability [34] of multidimensional problems.

Recently, it has been shown that Delta Test (DT) can be a powerful tool to determine the quality of a subset of variables by estimating the variance of the noise at the output [35]. Several studies related to feature selection using the DT have been developed using both local search strategies such as forward search (FS) [36] and forward-backward selection (FBS) [37, 38] and also global search techniques like tabu search [37, 39] and Genetic Algorithm (GA) [37]. Global search methods have the advantage of being able to escape from local minima, to which local methods are prone to converge [40].

This paper presents a global search technique based on GAs that manages not only to select, but also scale and project the input variables in order to minimize the DT criterion. The computation of the DT has been accelerated by using the approximate *k*-nearest neighbor approach [41]. The designed method makes use of MATLAB's Genetic Algorithm and Direct Search toolbox for the GA-based search. This methodology can be generalized to all regression problems. In this study we are going to focus mainly on time series processing, but we have also included non-time series datasets to show that the methodology can be generalized to all regression problems regardless of their nature. The predictive study of the time series is not going to be addressed, as the methodology only provides reduced datasets for their later modeling.

This paper is organized as follows: Section 2 explains the DT criterion and its role in the present study. Section 3 presents the approximate *k*-nearest neighbors algorithm that has been used to speed up the DT calculation. Section 4 introduces the motivation for the GA and the fitness function optimization methods, such as scaling, projection and their variations with a fixed number of variables. Section 5 describes the datasets tested, preprocessing and hardware/software specifications for the performed experiments, and finally, Section 6 discusses the most relevant results obtained.

2. Delta Test

Delta Test was introduced by Pi and Peterson for time series [42] and recently further analyzed by Liitiäinen *et al.* [43]. However, its applicability to variable selection was proposed in [35]. The DT is a nonparametric noise estimator, i.e. it aims to estimate the variance of the noise at the output, or the mean squared error (MSE) that can be achieved without overfitting. Given N input-output pairs $(\vec{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$, the relationship between $\vec{x}_i$ and y_i can be expressed as

$$y_i = f(\vec{x}_i) + \eta_i, \qquad i = 1, ..., N \,, \tag{1}$$

where f is the unknown function and η is the noise. The DT estimates the variance of the noise η.

The DT is useful for evaluating the nonlinear correlation between input and output variables. According to the DT, the selected set of input variables is the one that represents the relationship between the input variables and the output variable in the most deterministic way.

The DT is based on the hypothesis of continuity of the regression function. If two points $\vec{x}_1$ and $\vec{x}_2$ are close in the input variable space, the continuity of regression function implies that the outputs $f(\vec{x}_1)$ and $f(\vec{x}_2)$ will be close enough in the output space. If this is not accomplished, it is due to the influence of the noise.

The DT can be interpreted as a particularization of the Gamma Test [44] considering only the first nearest neighbor. This yields a fully nonparametric method as it removes the only hyperparameter (number of neighbors) that had to be chosen for the Gamma Test. Let us denote the nearest neighbor of a point $\vec{x}_i \in \mathbb{R}^d$ as $\vec{x}_{NN(i)}$. The nearest neighbor formulation of the DT estimates $\mathrm{Var}[\eta]$ by

$$\mathrm{Var}[\eta] \approx \delta = \frac{1}{2N} \sum_{i=1}^{N} (y_i - y_{NN(i)})^2 \,, \tag{2}$$

where $y_{NN(i)}$ is determined from the input-output pair $(\vec{x}_{NN(i)}, y_{NN(i)})$. For a proof of convergence the reader should refer to [44].

3. Approximate *k*-nearest neighbors

Nearest neighbor search is an optimization technique for finding closest points in metric spaces. Specifically, given a set of n reference points R and query point q, both in the same metric space V, we are interested in finding the closest or nearest point $c \in R$ to q. Usually, V is a d-dimensional space $\mathbb{R}^d$, where distances are measured using Minkowski metrics (e.g. Euclidean distance, Manhattan distance, max distance).

The simplest solution to this neighbor search problem is to compute the distance from the query point to every other point in the reference set, while registering and updating the position of the nearest or *k*-nearest neighbors of every point. This algorithm, sometimes referred to as the naive approach or brute-force approach, works for small datasets, but quickly becomes intractable as either the size or the dimensionality of the problem becomes large, because the running time is $O(dn)$. In practice, computing exact nearest neighbors in dimensions much higher than 8 seems to be a very difficult task [41].

Few methods allow to find the nearest neighbor in less time than the brute-force computation of all distances does. In 1977,

Friedman *et al.* [45] showed that $O(n)$ space and $O(\log n)$ query time are achievable through the use of *kd*-trees. However, even these methods suffer as dimension increases. The constant factors hidden in the asymptotic running time grow at least as fast as 2^d (depending on the metric).

In some applications it may be acceptable to retrieve a "good guess" of the nearest neighbor. In those cases one may use an algorithm which does not guarantee to return the actual nearest neighbor in every case, in return for improved speed or memory saving. Such an algorithm will find the nearest neighbor in the majority of cases, but this depends strongly on the dataset being queried. It has been shown [41] that by computing nearest neighbors approximately, it is possible to achieve significantly faster running times (on the order of tens to hundreds) often with relatively small actual errors.

The authors [41] state that given any positive real ϵ, a data point p is a $(1+\epsilon)$-approximate nearest neighbor of q if its distance from q is within a factor of $(1+\epsilon)$ of the distance to the true nearest neighbor. It is possible to preprocess a set of n points in $\mathbb{R}^d$ in $O(dn \log n)$ time and $O(dn)$ space, so that given a query point $q \in \mathbb{R}^d$, and $\epsilon > 0$, a $(1+\epsilon)$-approximate nearest neighbor of q can be computed in $O(c_{d,\epsilon} \log n)$ time, where $c_{d,\epsilon} \leq d\lceil 1+6d/\epsilon \rceil^d$ is a factor depending only on dimension and ϵ. In general, it is shown that given an integer $k \geq 1$, $(1+\epsilon)$ approximations to the k-nearest neighbors of q can be computed in additional $O(kd \log n)$ time.

This faster neighbor search has been applied to the computation of the DT as expressed in Eq. 2 with high computational savings.

4. Using genetic algorithms for global search

To date, the GA has been successfully applied for variable selection in many publications [37, 46, 47, 48, 49, 50]. Their success stems from the fact that they manage to carry out a global optimization of the selected set of variables. In such a setting, convergence of GA will depend on the available time, but the diversity of solutions produced in each generation allows the search to reach good solutions.

The purpose of the GA in this work is the global optimization of the scaling weights and projection matrix that minimize the DT when applied to datasets built from time series data, although this approach may apply to other regression problems. This study intends to find the optimal DT value in a fixed number of generations. Pure selection (i.e. assigning only '0' or '1' scaling factors to each variable) would clearly outperform scaling in terms of speed, but the best DT found is often suboptimal. Scaling or projection are necessary to get closer to the optimal set of solutions. For that reason, a real-coded GA is proposed to optimize a population of chromosomes that encode arrays of potential solutions. The following subsections describe the different types of variable preprocessing: scaling, scaling + projection, and their corresponding versions with fixed number of variables.

4.1. Scaling (S)

The target of performing scaling is to optimize the value of the DT beyond the minimum value that can be obtained with pure selection. When performing scaling, the variables are weighted according to their influence on the output variable. Let us consider f as the unknown function that determines the relationship between the N input-output pairs of a regression problem, $y = f(\vec{x}) + \eta$, with $\vec{x} \in \mathbb{R}^d, y \in \mathbb{R}$ and $\eta \in \mathbb{R}$ is a random variable that represents the noise. Thus, the estimate of the output, $\hat{y} \in \mathbb{R}$, can be expressed as $\hat{y} = g(\vec{x}^s)$, where $\vec{x}^s \in \mathbb{R}^d$ is the modified sample with scaling weights $\vec{s} = [s_1, s_2, ..., s_d]$ and g is the model that best approximates the function f. The objective is to find a scaling vector $\vec{s} \in \mathbb{R}^d$ such that

$$\hat{y} = g\left(s_1 x_1, s_2 x_2, \ldots, s_d x_d\right) \tag{3}$$

minimizes Var$[\eta]$ for the given problem.

In the existing variable selection literature there are several applications of scaling to minimize the DT, but often keeping a discrete number of weights [36, 37, 38] instead of using unconstrained real values like in this study. In each generation of the GA, every input sample $\vec{x}_i = [x_{i1}, x_{i2}, \ldots x_{id}]$ from the dataset $X_{[N\times d]}$ is multiplied element by element by an individual $\vec{s}$ (array of scaling factors), forming a new dataset $X^s_{[N\times d]}$:

$$x^s_{ij} = s_j x_{ij}, \qquad i = 1, \ldots, N, \;\; j = 1, \ldots, d \; . \tag{4}$$

Thus, for a population of p individuals the same number p new datasets will be created. The DT is calculated by obtaining the Euclidean distances among the weighted input samples X^s. After composing the new dataset X^s, the first approximate nearest neighbor of each point is selected using the method described in Section 3 and the DT is obtained from the difference between their corresponding outputs, according to Eq. 2. When a predefined number of generations has been evaluated, the GA returns the fittest individual and its corresponding DT value.

4.2. Scaling + projection to k dimensions (SP-k)

A projection can be used to reduce the number of variables by applying a linear (idempotent) transformation, represented by a matrix $P_{[d\times k]}$, to the matrix of input samples $X_{[N\times d]}$, resulting in a lower dimensional matrix $X^p_{[N\times k]}, k < d$:

$$X^p_{[N\times k]} = X_{[N\times d]}\, P_{[d\times k]} \; . \tag{5}$$

Although it might seem counterproductive, the idea of the developed method that combines scaling and projection is to add new variables to the input space (the projection of the input vectors on k dimensions). Eq. 6 describes how these new variables are attached to the input matrix as new columns:

$$X^{sp}_{[N\times(d+k)]} = [X^s_{[N\times d]}, X^p_{[N\times k]}] \; , \tag{6}$$

where X^s is the scaled version of X as calculated in Eq. 4, X^p is the projected version of X and X^{sp} is the new scaled/projected input matrix. In this case, the length of the chromosome increases linearly with parameter k, indicating that this value should be kept low to attain reasonable running times.

With a combination of both scaling and projection, the optimization problem should be able to reach a DT value that is not larger than the value obtained for scaling or projection alone. Consider the following two special cases. In the first special case, projection columns are set to zero values $X^{sp} = [X^s, \mathbf{0}_{[N\times k]}]$. This special case is just a scaling problem with additional zero columns that do not influence the search process, but only increase computational time. The second special case is similar, with all elements of X^s set to zero, i.e. $X^{sp} = [\mathbf{0}_{[N\times d]}, X^p]$, leading to a pure projection problem with extra computational cost. These two extreme cases suggest that by allowing both X^s and X^p to have real values, it becomes possible to find solutions that are at least as good as solutions for either scaling or projection problem.

4.3. Scaling with a fixed number of variables

In many real world datasets, the number of samples is sometimes so large ($N > 10000$) that optimizing scaling weights takes a considerable amount of time. This is due to the high computational cost of the inherent nearest neighbor search in the DT formula. One approach to solve this would simply be to randomly discard some portion of the samples in order to speed up calculation time, but there is risk of losing valuable data and there is no clear method to select important samples. Instead of removing samples, a different strategy involves drastically reducing the number of variables by forcing most of the scaling weights to have zero value ($s_i = 0$). To achieve this goal, an additional constraint is added to the problem which requires that at most d_f scaling weights have non-zero values. Therefore, d_f variables are fixed to be included in final scaling vector $\vec{s}$ and the remaining $d-d_f$ weights are forced to zero which effectively changes the dimensionality of the dataset. The computation of nearest neighbor search is reduced to a lower d_f-dimensional space. Thus, the fixed method enables a quick insight into the d_f most relevant variables of the regression problem.

The parameter d_f should not be considered as an additional hyperparameter to the problem, since the optimization with restricted number of scaling weights gives larger DT values compared to optimization without any restrictions. If we consider the scaling without any additional constraints (optimization of all d variables), we should be able to reach the global minimum, since it is included in the search space. Removing any of the variables carries the risk of excluding the global minimum from the search space (unless those variables have zero weights in the global minimum), leading to solutions with larger DT values. Thus, to find global minimum one should set $d_f = d$. However, the search for nearest neighbors is computationally more expensive in d-dimensional space than in d_f-dimensional one. Introducing d_f parameter enables the control over the trade-off between the DT values and computational time.

For easier notation and understanding, we refer to standard scaling as *scaling* or *pure scaling*, while scaling with the fixed number of variables is referred to as *fixed scaling*. The same setup of the GA can be used for both scaling problems. For the fixed scaling problem, one can just take the d_f most important or largest weights, effectively performing a ranking of scaling weights. On the other hand, the chromosomes can be fixed to have at most d_f non-zero values. With modified chromosomes, the crossover and mutation operators have to be modified accordingly to further preserve the required constraint. However, both approaches tend to converge extremely quickly to suboptimal solutions in just a couple of tens of generations. A different approach would be to consider this as a multi-objective (MO) optimization problem [51, 52, 53], where one objective is the minimization of the DT, the main goal, and the other objective is the absolute difference between the number of non-zero scaling weights and the desired value d_f, i.e.

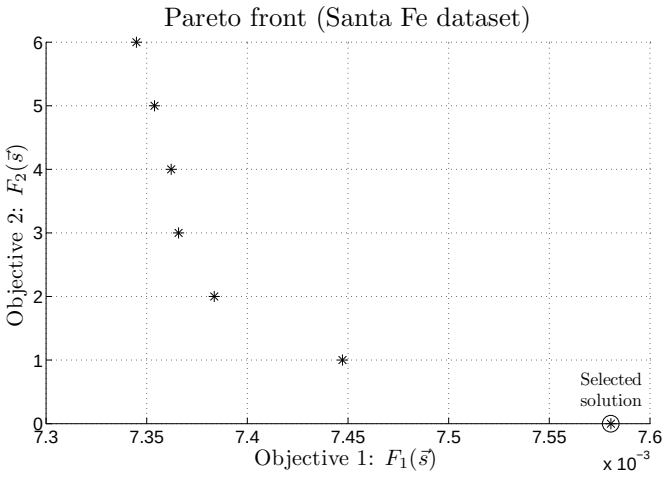


Figure 1: Pareto front for Santa Fe dataset, using scaling with fixed number of variables and $d_f = d/2$. The desired solution is the one that, in first place, adopts the desired number of fixed variables d_f (or as close as possible) and, in second place, minimizes the DT with that constraint.

$$F_1(\vec{s}) = \mathrm{Var}[\eta] \text{ on scaled dataset } X^s \tag{7}$$

$$F_2(\vec{s}) = \left| d_f - |\{s_i \neq 0 \mid i = 1, \ldots, d\}| \right|. \tag{8}$$

MO optimization tries to find the Pareto Optimal Front [54] (a set of non-dominated solutions) instead of a single solution. This set contains solutions where the values of objective functions are in *conflict*, i.e. improving one objective leads to deterioration in the other objective(s). Therefore, the result to a MO problem is a set of solutions on different pareto fronts, after which the user selects one (or more) based on his/her preference. In this study, when the solutions are returned, we look for the one with the exact required number d_f of non-zero scaling weights and the smallest DT. If such a solution does not exist, the one with the lowest F_2 value is used, that is, we try to stay close to d_f variables. A pareto front of Santa Fe dataset (see Table1 for details) is shown in Figure 1 to illustrate this.

The algorithm used for MO optimization is the Elitist Non-Dominated Sorting Genetic Algorithm proposed in [51], denoted NSGA-II. It works by constructing the new population layer by layer of non-dominated fronts. To ensure that the population has always the same size, the last layer to be added has to be split up into two parts. The part that is included in the next population contains solutions from the least crowded area of that front. This crowded comparison is based on *crowding distance*, which is computed in the objective function space. For

details see [51]. The overall complexity of NSGA-II is $O(hp^2)$, where h is the number of objectives (in our case $h = 2$) and p is the size of the population.

Fixed scaling is easily extended to include the projection problem, in the same manner as explained in Section 4.2 for scaling + projection. The combination of scaling with a fixed number of variables and projection will be referred to as *fixed scaling + projection*. The projection in this problem is not modified, only the scaling is replaced with the fixed version.

5. Experiments

The experiments were carried out using MATLAB R2009a (TheMathworks Inc., Natick, MA, USA) and its Genetic Algorithm and Direct Search Toolbox (GADS). Creation, crossover and mutation operators are implemented outside of the toolbox. The approximate nearest neighbor search uses a C++ library available at [55]. The hardware platform used was an Intel Core i7™TM 920 processor (CPU clock: 4.2 GHz, Cache size: 8 MB) with 6 GB of system memory running Windows 7 (64-bit).

The populations are initially created using a specific function that assigns a uniform initialization to a percentage of the population and the rest can be customized by the user, specifying how many of the remaining individuals are initialized randomly and how many of them are left as zeros. The function is flexible in the sense that the desired percentage of the initial population can be further split into more subsets, each one with a customizable percentage of randomly initialized individuals.

The crossover and mutation operators have also been implemented as custom functions according to [37]. The mutation operator is a pure random uniform function that operates at a gene level. A relatively high mutation rate (0.1) is used to enable an effective exploration of the solution space. The crossover operator is BLX-α [56], that is specifically developed for a real-coded GA. BLX-α consists in, given two individuals $I_1 = (i_1^1, i_2^1, ...i_d^1)$ and $I_2 = (i_1^2, i_2^2, ...i_d^2)$ with $(i \in \mathbb{R})$, a new offspring $O = (o_1, ..., o_j, ..., o_d)$ can be generated where $o_j, j = 1, ..., d$ is a random value chosen from a uniform distribution within the interval $[i_{min} - \alpha \cdot B, i_{max} + \alpha \cdot B]$ where $i_{min} = \min(i_j^1, i_j^2)$, $i_{max} = \max(i_j^1, i_j^2)$, $B = i_{max} - i_{min}$ and $\alpha \in \mathbb{R}$. The selection operator is the binary tournament selection [57]. The binary tournament does not require the computation of any probability for each individual, saving a considerable amount of operations in each iteration. This can increase computation time in problems that require large number of individuals in the population.

The population size was fixed to 150 individuals. This value appears to be a good compromise between performance and computational cost for GA-based search applied to similarly sized datasets [37, 58]. After some preliminary tests, the number of generations was fixed to 200 to ensure convergence in all cases. An alternative to this setting could be allowing a dynamic management of the number of generations to be evaluated before stopping the search. Thus, the search would be stopped if no improvement in the DT has been found for n generations, indicating that the algorithm has converged.

The fitness function of the GA is the DT computed for different types of problems. In the MO optimization, the DT is one of two objective functions. In this paper, we denote with DTS the optimization of scaling problem using DT, with DTFS the fixed scaling problem, with DTSP-k the problem of scaling + projection to k dimensions, with DTFS-d_f the fixed scaling with d_f variables and finally DTFSP-d_f-k is the problem of fixed scaling with d_f variables plus projection to k dimensions. To emphasize the goodness of these methods, pure selection (DTSL) has also been included in the comparison.

From previous analysis [37, 58], the best crossover and mutation rates for a feature selection application using the GA are 0.85 and 0.1 respectively, and an elitism of 10% of the individuals was the best compromise.

To sum up, the GA parameters were set as follows:

- Number of averaged runs: 10
- Number of generations evaluated: 200
- Population size: 150
- Population initialization: 20% uniform / 80% custom. The customized part is further divided into three parts:
  - 1/3 with 90% zeros and 10% random genes
  - 1/3 with 80% zeros and 20% random genes
  - 1/3 with 70% zeros and 30% random genes
- Crossover operator: BLX-α ($\alpha = 0.5$)
- Selection function: Binary tournament
- Crossover rate: 0.85
- Mutation rate: 0.1[1]
- Elitism: 10%
- Mutation function: Random uniform

The parameter d_f was set to $\lceil d/2 \rceil$ for all datasets throughout the experiments.

5.1. Datasets

The described methods have been evaluated on eight time series and two standard regression datasets, to show the applicability of this methodology to generic regression problems. For the time series, the data matrices were composed using one-step-ahead direct prediction strategy [59, 60]. The size of the built datasets are listed in Table 1. For the time series, the number of variables refers to the regressor size, which was chosen according to the periodicity of each series.

Some of the series were preprocessed in order to make them more stationary by removing the trend and seasonality. This is particularly the case for all of the three series of ESTSP 2008 competition. The first series (ESTSP 2008a) is actually a set of three series, with two exogenous and one target series. The goal is to predict the next values of the target series. For our

[1] This is the same value used by Oh *et al.* [47] and Guillén *et al.* [37], also for feature selection. The rate is higher than usual to enable a thorough exploration of all regions of the solution space, rather than focus on the refinement of a small number of candidate solutions.

Table 1: Datasets tested

Dataset	Instances	Variables
Mackey-Glass 17 [61]	1500	20
Mackey-Glass 30 [61]	1500	20
Poland electricity [62]	1400	12
Santa Fe [62]	1000	12
ESTSP 2007 [62]	875	55
ESTSP 2008a [62]	354	20
ESTSP 2008b [62]	1300	15
Darwin SLP [63]	1400	13
Housing [64]	506	12
Tecator*[65]	215	100

* This dataset was normalized in a sample-wise way instead of the typical variable-wise way, because it has been proved that better DT values are achieved with this variation [58].

experiments we only used the target series, as correlation of the target series with the other two exogenous series is not significant. The target series is then transformed by taking the first order difference. ESTSP 2008b series was transformed by applying $x'_t = log(x_t/x_{t-1})$ to the original series defined by $x_t,\ t \in [1, \ldots, 1300]$, in order to remove the trend.

All datasets were normalized to zero mean and unit variance to prevent variables with high variance from dominating over those with a lower one. Therefore, all DT values shown in this paper are normalized by the variance of its respective output variable. No splitting of datasets was performed (i.e. training and test sets) because the objective of the paper is the data preprocessing to minimize the DT value and not the fitting of a model to the data. It is important to note that in a real-world prediction application, the methodology should be applied to the training data available, but in these experiments the method was applied to the full datasets to provide repeatability, independently of the particular choice of training and test subsets.

6. Results

6.1. Approximate k-nearest neighbor performance

A preliminary test was carried out to assess the speed of the approximate method of nearest neighbor computation as a function of ϵ. Typically, ϵ controls the trade-off between efficiency and accuracy. When ϵ is set larger, the approximation is less accurate, and the search completes faster.

The averaged results (10 runs) for Santa Fe, ESTSP 2007 and Mackey Glass 17 datasets, after 200 generations, are shown in Figure 2 for values of ϵ between 0 and 2. Higher values were not tested as the DT estimate quickly deteriorates. The approximate method proves to be faster for larger values of ϵ, as expected. The computational time improvement for $\epsilon = 1$ with respect to exact k-NN ($\epsilon = 0$) ranges from 15% (Santa Fe) to 28% (ESTSP 2007).

Additionally, a study on the DT performance has been done, using several values of ϵ for approximate k-NN. The DT performance is plotted in Figure 3 using DTSP-1. A degradation of the DT is expected when allowing higher error bound. Experimental results show a clear increase trend for ESTSP 2007 and Mackey Glass 17 datasets when $\epsilon > 1$. From these observations, we select the value $\epsilon = 1$ for the rest of the experiments as it seems the best compromise between speed-up and goodness of the DT estimate. The DT given by the approximate search with $\epsilon = 1$ stays close to the value given by the exact search in the majority of cases. The greatest difference was registered for Santa Fe, where there is an 8% DT reduction. Thus, we are reducing the computational cost with no significative performance loss. This result makes the approximate k-NN approach very interesting, especially for large datasets.

6.2. DT performance

The average DT values computed by each method for each dataset are shown in Figure 4. The scaling and projection factors assigned to each variable have been omitted because the subject of interest are the DT values only.

From the inspection of the results, one can affirm that all the proposed methods outperform pure selection in the considered scenarios, except from rare exceptions given by the fixed methods. The best performing method is DTSP-1 in most cases, followed by DTFSP. However, the DTFSP method ties with DTSP in some datasets, such as the Mackey Glass series. Overall, the methods that include projections are the most advantageous. This was an expected result because they include the benefits of scaling and also leverage the possibility of using newly created variables to gain an advantage over scaling alone.

In general, the fixed variations provide slightly worse DT values than their standard counterparts (e.g. Santa Fe, Poland electricity), meaning that learning models will be able to give similar performance on the halved dataset. Since only half of the variables are used, the training times of models will greatly benefit from this reduction. The fixed version also gives an insight into the most relevant variables, and in these experiments the $\lceil d/2 \rceil$ most important dimensions for prediction. For ESTSP datasets, values of DTFS are not on the same level as those of DTS, suggesting that more than $\lceil d/2 \rceil$ variables are required for better prediction.

6.3. Computational time

A computational time comparison of the presented methods is shown in Figure 5. The pure selection is obviously the fastest method because the combination of solutions to try is very limited. It is noticeable that DTSP-1 requires similar computational time to perform the 200 generations to DTS, and in some cases (ESTSP 2007, Darwin SLP) even improving times computed for pure selection. This might seem contradictory, as the size of the individuals is twice the size of those used for DTS in the GA setup. Although the computational time for GA doubles when moving from DTS to DTSP-1, the running time of DT optimization is dominated by nearest neighbor search. The faster calculation time for DTSP-1 could be attributed to the

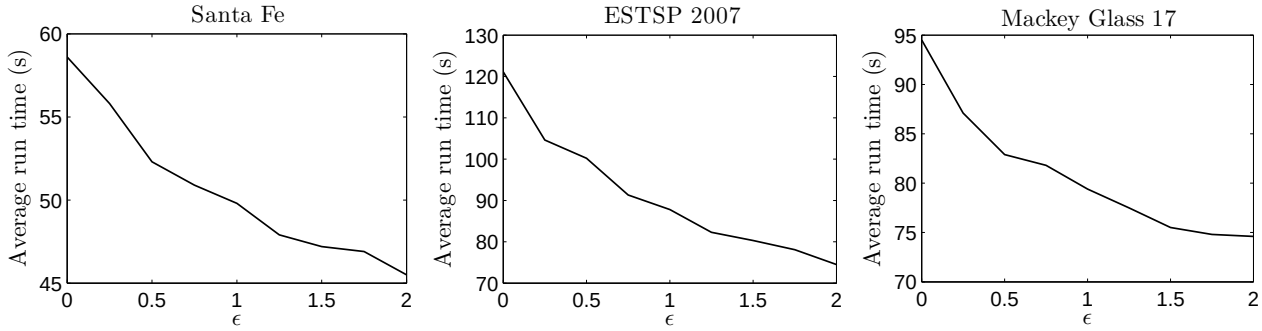


Figure 2: Computational time of the approximate k-NN search as a function of ϵ for three datasets. The results were obtained running DTSP-1 for 200 generations and 10 runs were averaged.

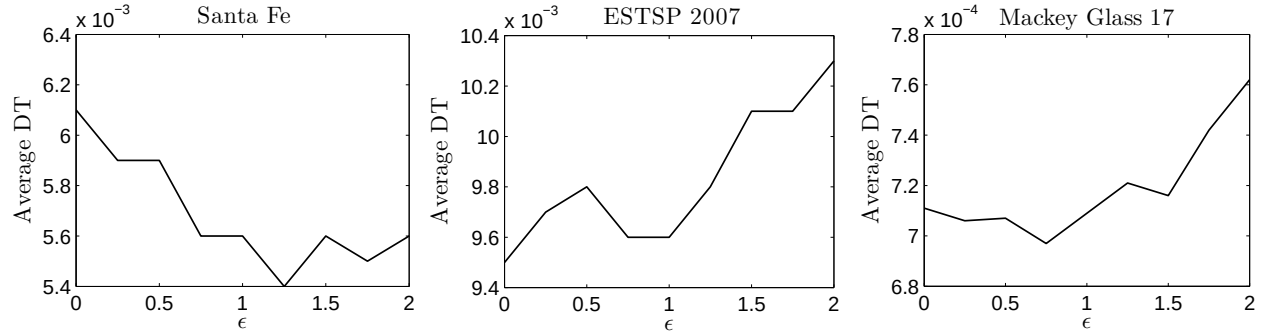


Figure 3: DT performance comparison for approximate k-NN search and different values of ϵ for three datasets. The results were obtained running DTSP-1 for 200 generations and 10 runs were averaged.

construction of the underlying data structure of approximate nearest neighbors, which uses a hierarchial space decomposition tree called balanced-box decomposition (BBD) tree [41]. Additional dimensions might lead to favorable splitting of the points/samples into leaves of the tree, eventually improving response time for query searches.

The fixed versions yielded good results in terms of DT values for some datasets, but their computational times are generally higher than their non-fixed versions. When using MO optimization there is an additional cost inherent in NSGA-II method, which computes crowding distance to ensure that individuals from least crowded areas are included in the next population. The additional $O(hp^2)$ complexity of NSGA-II for $p = 150$ slightly increases the running time for most datasets. The only exceptions is ESTSP 2008b, where the computational time for fixed methods is lower than for DTS/DTSP-1.

In the next subsection, the computational time and performance of the DTSP method is further analyzed for several values of k.

6.4. Projection to $k > 1$ dimensions

From the previous results one can extract the conclusion that DTSP-1 has clearly outperformed the rest of the methods while still keeping reasonably low computational times in many scenarios (except for ESTSP 2008b, Darwin SLP and Poland electricity series). The DTSP method also offers the possibility of projecting to more than one dimension. In this subsection, projections to $k = \{1, 2, 3, 4, 5\}$ dimensions are tested for each time series dataset. Figure 6 illustrates the DT results obtained and Figure 7 represents the computational time evolution.

By looking at the results it is easy to observe that, for all datasets, the value of DT has an optimum value after which it starts to rise again when adding more projections. The question that remains is how to automatically select a good value for k that optimizes the DT. One possible heuristic is the following: Start with $k = 1$ and compute the DT criterion. Then progressively increase k by 1 until the value of DT no longer improves. Since the result of the GA depends on the initial population, in the proposed heuristic, instead of taking only one run of the GA, several runs should be performed and the minimum taken as the result for a certain value of k. The downsides of this approach are: a) the huge computational cost, since for each value of k, the GA is applied several times to produce reliable DT values, and b) the possibility of stopping prematurely due to local minima.

The computational times show a general increasing tendency, which is logical due to the complexity increase with k. The only exception was ESTSP 2007, for which computational times show an irregular descending behavior. For Darwin SLP, the computational time registered for $k = 1$ is slightly higher than expected, as compared to the time required to project to more dimensions.

Table 2: Minimum DT values calculated for the ten datasets using FBS, tabu search and DTSP-k (denormalized values in brackets)

Dataset	FBS		Tabu		DTSP-k	
	Selection	Scaling	Selection	Scaling	k	DT
Santa Fe	0.0164 (36.09)	0.0094 (20.71)	0.0164 (36.11)	0.0108 (23.82)	1	0.0050 (11.00)
ESTSP 2007	0.0133 (0.082)	0.0137 (0.084)	0.0135 (0.083)	0.0156 (0.096)	1	0.0092 (0.056)
ESTSP 2008a	0.4959 (4.978)	0.4553 (4.570)	0.5409 (5.430)	0.4028 (4.043)	5	0.2223 (2.238)
ESTSP 2008b	0.2907 (9.7E-3)	0.2775 (9.3E-3)	0.2907 (9.7E-3)	0.2782 (9.3E-3)	3	0.1878 (6.3E-3)
Darwin SLP	0.1062 (0.7214)	0.1019 (0.6923)	0.1071 (0.7277)	0.0982 (0.6671)	4	0.0725 (0.4925)
Mackey Glass 17	11.5E-4 (5.9E-5)	9.5E-4 (4.9E-5)	11.5E-4 (5.9E-5)	9.5E-4 (4.9E-5)	2	6.3E-4 (3.2E-5)
Mackey Glass 30	4.0E-3 (3.2E-4)	3.8E-3 (3.0E-4)	4.0E-3 (3.2E-4)	3.8E-3 (3.0E-4)	1,2	1.6E-3 (1.3E-4)
Poland electricity	0.0481 (0.0013)	0.0404 (0.0011)	0.0481 (0.0013)	0.0379 (0.0010)	4	0.024 (6.6E-4)
Housing	0.0710 (6.009)	0.05654 (4.783)	0.0720 (6.089)	0.0558 (4.720)	3	0.0385 (3.257)
Tecator	0.0136 (2.2076)	0.0149 (2.412)	0.0112 (1.825)	0.0248 (4.023)	2,3	0.0028 (0.454)

Finally, we have compared the minimum DT values achieved with DTSP-k to some popular search methods that can use DT as performance criterion, such as forward-backward selection and tabu search with the settings of [37], both for selection and discrete scaling (with 10 equally spaced levels). For a fair comparison, these methods were executed in the same machine as DTSP-k and the the same amount of solutions were evaluated by each method. The results are listed in Table 2. The proposed methodology easily outperforms the classic techniques by a large margin.

6.5. Discussion

After extensive testing, several conclusions can be drawn. We have profited from using the approximate version of the nearest neighbor computation to evaluate the DT. Preliminary tests on three datasets have shown that the approximate k-NN algorithm produces computational time improvements of between 15% and 28% while keeping very similar DT values to the ones obtained by the exact search. Better improvements can be obtained at the expense of higher DT. This trade-off is controlled by parameter ϵ, which provided acceptable DT results in the range [0,1]. The results obtained suggest that the approximate method is suitable for speed critical applications or large datasets.

With respect to the DT performance of the different fitness functions, all the proposed variations provide better results than selection. The improvement introduced depends on the method used, but generally DTSP provides the best results, followed by DTFSP. These improvements range from 24% (Darwin SLP) to almost 70% (Santa Fe, Tecator). The importance of projection stands out especially in Tecator dataset. Where scaling is hardly able to obtain any improvement against selection, projection manages to reduce the estimate by a great amount. In some cases, fixed methods do not perform well (e.g. most ESTSP time series). This is probably because in complex problems, the limited amount of variables ($d/2$) did not allow further DT minimization.

We found that the DTSP method was not only the best overall performer, but it also generally reached these good results in a reasonable amount of time. It performs comparably to DTS in terms of speed, despite having twice the number of genes to optimize. In the particular case of ESTSP 2007 it outperformed pure selection in terms of speed, which is a remarkable result.

The adjustable number of projections also aided in the obtention of lower DT values. The progression of the DT curves as a function of k shows a minimum where the optimum DT has been registered. As we only tested projections to $k \leq 5$ dimensions, better values may be found for higher k's, at the expense of computational time. The increase in computational time as a function of k generally follows an increase with k, save ESTSP 2007 dataset, which produced an irregular descending trend.

The second overall best performing method has been DTFSP. The improvement provided by this method follows a similar trend to DTSP, as both include projection capabilities. Likewise, DTFS was able to show similar performance to DTS using only half of the variables, which is a promising result. However, the running times for DTFS and DTFSP were higher than for DTS and DTSP, respectively, in 8 out of 10 datasets. The fact of carrying out a double objective optimization without increasing the number of generations could have affected the performance. Besides, we believe that the strong constraint imposed by grounding half the regressor variables to zero can justify this difference. Of course, the number of variables that are considered zero can be tweaked to match the needs of every particular user (e.g. for applications limited by number of variables). To sum up, the fixed methods can be very beneficial when building a model because it will only be necessary to deal with a fraction of the initial number of inputs, achieving similar results.

Casting the scaling problem with a fixed number of variables into MO setting increases the run time on the tested datasets. In order to achieve lower running times, the number of individuals in the population has to be reduced, which influences the exploration capabilities of the GA in a negative way. The additional computational time of NSGA-II prevents it from being used in this type of problem. Therefore, faster and simpler techniques should be employed to lower the running times of the fixed scaling (plus projection) problem. One such possibility is the island GA with migration policies that do not have

such high complexity.

7. Conclusion

This paper has presented a fast methodology for DT minimization based on a global search guided by a GA, which has been successfully applied to a variety of time series datasets. The methodology can be generalized to other regression problems with a single output, as it has also been shown. The most important goals of the proposed methodology are to reduce the datasets in order to simplify the complexity of the necessary modeling schemes and to improve interpretability by pruning unnecessary variables, only keeping those that are really important for the prediction.

The DT method requires the computation of the nearest neighbor of each point. The time needed for this computation has been greatly alleviated by using an approximate version of the *k*-nearest neighbor algorithm. The DT was optimized using several methodologies: scaling of variables (DTS), scaling + projection to a number of dimensions (DTSP) and versions of these with a fixed number of variables (DTFS and DTFSP, respectively). These methods can minimize the DT beyond the limits imposed by pure selection and help to increase interpretability of the datasets.

The results obtained are very promising, especially for the methods that include projection. Projection has helped to bring out underlying relationships between variables that were initially not apparent, thus allowing a significant DT minimization with respect to scaling alone. DTSP was the best performing method in all scenarios and it reached a maximum DT minimization of 70% over pure selection. Moreover, the low computational time of this method makes it suitable for problems that involve large datasets. The possibility of varying the number of dimensions to project to enables a fine refinement of the result that proved useful in all tests.

References

[1] G. Box, G. Jenkins, G. Reinsel, Time Series Analysis: Forecasting and Control, 3rd Edition, Prentice-Hall, Englewood Cliffs, NJ, 1994.

[2] P. Brockwell, R. Davis, Introduction to Time Series and Forecasting, 2nd Edition, Springer, Berlin, 2002.

[3] M. Casdagli, Nonlinear prediction of chaotic time series, Physica D 35 (1989) 335–356.

[4] M. Clements, P. Franses, N. Swanson, Forecasting economic and financial time-series with non-linear models, International Journal of Forecasting 20 (2) (2004) 169–183.

[5] T. Ozaki, Nonlinear time series models and dynamical systems, Time Series in the Time Domain (1985) 2583.

[6] M. Priestley, Non-Linear and Non-stationary Time Series Analysis, Academic Press, New York, 1988.

[7] T. Rao, M. Gabr, Introduction to bispectral analysis and bilinear time series models, in: Lecture Notes in Statistics, Vol. 24, Springer, 1984.

[8] D. Rumelhart, J. McCleland, Parallel Distributed Processing, Explorations in the Microstructure of Cognition, Vol. 1 and 2, MIT Press, Cambridge, MA, 1987.

[9] J. Hansen, R. Nelson, Forecasting and recombining time series components by using neural network, Computational Operational Research 50 (2003) 307–317.

[10] H. Hwarng, Insights into neural network forecasting time series corresponding to ARMA(p,q) structures, Omega 29 (2001) 273–289.

[11] V. Kodogiannis, A. Lolis, Forecasting financial time series using neural network and fuzzy system-based techniques, Neural computing & applications 11 (2) (2002) 90–102.

[12] Z. Tang, P. Fishwick, Feedforward neural nets as models for time series forecasting, ORSA J. Comput. 5 (1993) 374–385.

[13] A. Tawfiq, E.A.Ibrahim, Artificial neural networks as applied to long-term demand forecasting, Artificial Intelligence Engineering 13 (1999) 189–197.

[14] G. Zhang, E. Patuwo, M. Hu, A simulation study of artificial neural network for nonlinear time-series forecasting, Computational Operational Research 28 (2001) 381–396.

[15] G. Zhang, M. Qi, Neural network forecasting for seasonal and trend time series, European Journal of Operational Research 160 (2005) 501–514.

[16] M. Niranjan, V. Kadirkamanathan, A nonlinear model for time series prediction and signal interpolation, in: International Conference on Acoustics, Speech, and Signal Processing, ICASSP-91, 1991, pp. 1713–1716.

[17] L. Qu, Y. Chen, Z. Liu, Time series forecasting model with error correction by structure adaptive RBF neural network, in: The Sixth World Congress on Intelligent Control and Automation, WCICA 2006, Vol. 2, 2006.

[18] R. Zemouri, D. Racoceanu, N. Zerhouni, Recurrent radial basis function network for time-series prediction, Engineering Applications of Artificial Intelligence 16 (5-6) (2003) 453–463.

[19] S. Mukherjee, E. Osuna, F. Girosi, Nonlinear prediction of chaotic time series using support vector machines, in: Proceedings of the VII Workshop on Neural Networks for Signal Processing, NNSP 1997, 1997, pp. 511–520.

[20] K. Müller, A. Smola, G. Rätsch, B. Schökopf, J. Kohlmorgen, Advances in Kernel Methods - Support Vector Learning, MIT Press, Cambridge, MA, 1999, Ch. Using support vector machines for time series prediction, pp. 243–254.

[21] F. Tay, L. Cao, Application of support vector machines in financial time series forecasting, Omega 29 (2001) 309–17.

[22] T. Trafalis, H. Ince, Support vector machine for regression and applications to financial forecasting, in: Proceedings of the International Joint Conference on Neural Networks, IJCNN 2000, Vol. 6, 2000.

[23] T. Koskela, M. Varsta, J. Heikkonen, K. Kaski, Recurrent SOM with local linear models in time series prediction, in: 6th European Symposium on Artificial Neural Networks, D-facto Publications, 1998, pp. 167–172.

[24] G. Simon, A. Lendasse, M. Cottrell, J.-C. Fort, M. Verleysen, Time series forecasting: Obtaining long term trends with self-organizing maps, Pattern Recognition Letters 26 (12) (2005) 1795–1808.

[25] K. Chen, C. Wang, A hybrid SARIMA and support vector machines in forecasting the production values of the machinery industry in Taiwan, Expert Systems with Applications 32 (1) (2007) 254–264.

[26] Y. Chen, B. Yang, J. Dong, Time-series prediction using a local linear wavelet neural network, Neurocomputing 69 (4-6) (2006) 449–465.

[27] C. Lee, Y. Chiang, C. Shih, C. Tsai, Noisy time series prediction using m-estimator based robust radial basis function neural networks with growing and pruning techniques, Expert Systems With Applications 36 (3P1) (2009) 4717–4724.

[28] R. Singh, S. Balasundaram, Application of extreme learning machine method for time series analysis, Int. Jour. Int. Tech 2 (4) (2007) 256–262.

[29] A. Jain, A. Kumar, Hybrid neural network models for hydrologic time series forecasting, Applied Soft Computing Journal 7 (2) (2007) 585–592.

[30] L. See, S. Openshaw, Hybrid multi-model approach to river level forecasting, Hydrol. Sci. J. 45 (4) (2000) 523–536.

[31] O. Valenzuela, I. Rojas, F. Rojas, H. Pomares, L. Herrera, A. Guillén, L. Marquez, M. Pasadas, Hybridization of intelligent techniques and ARIMA models for time series prediction, Fuzzy Sets and Systems 159 (7) (2008) 821–845.

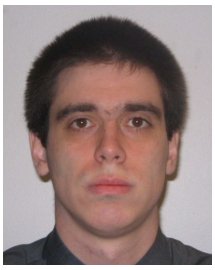
[32] L. Xiaoyu, W. Bing, Y. Simon, Time series prediction based on fuzzy principles, Tech. rep., Department of Electrical & Computer Engineering, FAMU-FSU College of Engineering, Florida State University, Tallahassee, FL.

[33] M. Verleysen, D. François, The curse of dimensionality in data mining and time series prediction, in: J. Cabestany, A. Prieto, F. Sandoval (Eds.), Lecture Notes in Computer Science, Vol. 3512, Springer, 2005, pp. 758–770.

[34] I. Guyon, S. Gunn, M. Nikravesh, A. Zadeh, Feature extraction: Foundations and applications (Studies in fuzziness and soft computing),

Springer-Verlag New York, Secaucus, NJ, USA, 2006.
[35] E. Eirola, E. Liitiäinen, A. Lendasse, F. Corona, M. Verleysen, Using the delta test for variable selection, in: Proc. of ESANN 2008, European Symposium on Artificial Neural Networks, Bruges, Belgium, 2008, pp. 25–30.
[36] Q. Yu, E. Séverin, A. Lendasse, A global methodology for variable selection: application to financial modeling, in: Proc. of MASHS 2007, ENST-Bretagne, France, 2007.
[37] A. Guillén, D. Sovilj, F. Mateo, I. Rojas, A. Lendasse, Minimizing the delta test for variable selection in regression problems, Int. J. of High Performance Systems Architecture 1 (4) (2008) 269–281.
[38] F. Mateo, A. Lendasse, A variable selection approach based on the delta test for extreme learning machine models, in: Proc. of ESTSP 2008, European Symposium on Time Series Prediction, Porvoo, Finland, 2008, pp. 57–66.
[39] D. Sovilj, A. Sorjamaa, Y. Miche, Tabu search with delta test for time series prediction using OP-KNN, in: Proc. of ESTSP 2008, European Symposium on Time Series Prediction, Porvoo, Finland, 2008, pp. 187–196.
[40] J. Holland, Adaptation in natural and artificial systems, University of Michigan Press, 1975.
[41] S. Arya, D. Mount, N. Netanyahu, R. Silverman, A. Wu, An optimal algorithm for approximate nearest neighbor searching fixed dimensions, Journal of the ACM (JACM) 45 (6) (1998) 891–923.
[42] H. Pi, C. Peterson, Finding the embedding dimension and variable dependencies in time series, Neural Computation 6 (3) (1994) 509–520.
[43] E. Liitiäinen, F. Corona, A. Lendasse, On nonparametric residual variance estimation, Neural Processing Letters 28 (3) (2008) 155–167.
[44] A. Jones, New tools in non-linear modelling and prediction, Computational Management Science 1 (2) (2004) 109–149.
[45] J. Friedman, J. Bentley, R. Finkel, An algorithm for finding best matches in logarithmic expected time, ACM Transactions on Mathematical Software (TOMS) 3 (3) (1977) 209–226.
[46] I.-S. Oh, J.-S. Lee, B.-R. Moon, Local search-embedded genetic algorithms for feature selection, Proc. of the 16th Int. Conference on Pattern Recognition 2 (2002) 148–151.
[47] I.-S. Oh, J.-S. Lee, B.-R. Moon, Hybrid genetic algorithms for feature selection, IEEE Trans. on Pattern Analysis and Machine Intelligence 26 (11) (2004) 1424–1437.
[48] W. Punch, E. Goodman, M. Pei, L. Chia-Shun, P. Hovland, R. Enbody, Further research on feature selection and classification using genetic algorithms, in: S. Forrest (Ed.), Proc. of the Fifth Int. Conf. on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, 1993, pp. 557–564.
[49] M. Raymer, W. Punch, E. Goodman, L. Kuhn, A. Jain, Dimensionality reduction using genetic algorithms, IEEE Trans. on Evolutionary Computation 4 (2) (2000) 164–171.
[50] Y. Saeys, I. Inza, P. Larranaga, A review of feature selection techniques in bioinformatics, Bioinformatics 23 (19) (2007) 2507–2517.
[51] K. Deb, S. Agrawal, A. Pratap, T. Meyarivan, A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II, Springer, 2000, pp. 849–858.
[52] K. Miettinen, Nonlinear Multiobjective Optimization, Vol. 12 of International Series in Operations Research and Management Science, Kluwer Academic Publishers, Dordrecht, 1999.
[53] R. Steuer, Multiple Criteria Optimization : Theory, Computation, and Application, Wiley, New York, Toronto, 1986.
[54] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, Wiley, Chichester, UK, 2001.
[55] http://www.cs.umd.edu/~mount/ANN/.
[56] L. Eshelman, J. Schaffer, Real-coded genetic algorithms and interval schemata, in: L. Darrell Whitley (Ed.), Foundation of Genetic Algorithms 2, Morgan-Kauffman Publishers, Inc., 1993, pp. 187–202.
[57] D. Goldberg, Genetic algorithms in search, optimization and machine learning, Addison Wesley, 1989.
[58] F. Mateo, D. Sovilj, R. Gadea, A. Lendasse, RCGA-S/RCGA-SP methods to minimize the delta test for regression tasks, in: J. C. et al. (Ed.), Lecture Notes in Computer Science, Vol. 5517, Springer, 2009, pp. 359–366.
[59] Y. Ji, J. Hao, N. Reyhani, A. Lendasse, Direct and recursive prediction of time series using mutual information selection, in: International Work-Conference on Artificial Neural Networks, IWANN, Springer, 2005, pp. 8–10.
[60] A. Sorjamaa, J. Hao, N. Reyhani, Y. Ji, A. Lendasse, Methodology for long-term prediction of time series, Neurocomputing 70 (16-18) (2007) 2861–2869.
[61] http://www.cse.ogi.edu/~ericwan/data.html.
[62] http://www.cis.hut.fi/projects/tsp/index.php?page=timeseries.
[63] http://www.stat.duke.edu/~mw/data-sets/ts_data/darwin.slp.
[64] http://archive.ics.uci.edu/ml/data sets/Housing.
[65] http://lib.stat.cmu.edu/data sets/tecator.

Fernando Mateo was born in Valencia, Spain, in 1981. He obtained his M.Sc. in Telecommunication Engineering in 2005 from the Universidad Politécnica de Valencia, Spain. Currently, he is a researcher pursuing his Ph.D. at the Institute of Applications of Information Technology and Advanced Communications, at the same University. In 2007 and 2008 he also collaborated with the Computer Science and Information Laboratory at Helsinki University of Technology, Finland. His research is related to machine learning methods and applications like position estimation in positron emission tomography detectors, prediction of contaminants in food and variable selection techniques for high-dimensional problems.



Dušan Sovilj obtained his B.Sc. in 2006 from University of Novi Sad, Serbia. He is pursuing a Master degree at Helsinki University of Technology in Finland, and at the same time working at the Time Series Prediction and Chemoinformatics Group at the same University. His main topics of research are time series prediction and variable selection in regression problems. He is also interested in artificial intelligence in computer games.

Rafael Gadea received the M.Sc. and Ph.D. degrees from the Universidad Politécnica de Valencia, Spain, in 1990 and 2000, respectively. Since 1992 he has been a lecturer of the Department of Electronics at the Universidad Politécnica de Valencia. Currently, he is assistant professor at Telecommunications Engineering School of the Universidad Politécnica de Valencia, Spain. His areas of research interest include hardware description languages, design of FPGA-based systems and design of neural networks and cellular automata.

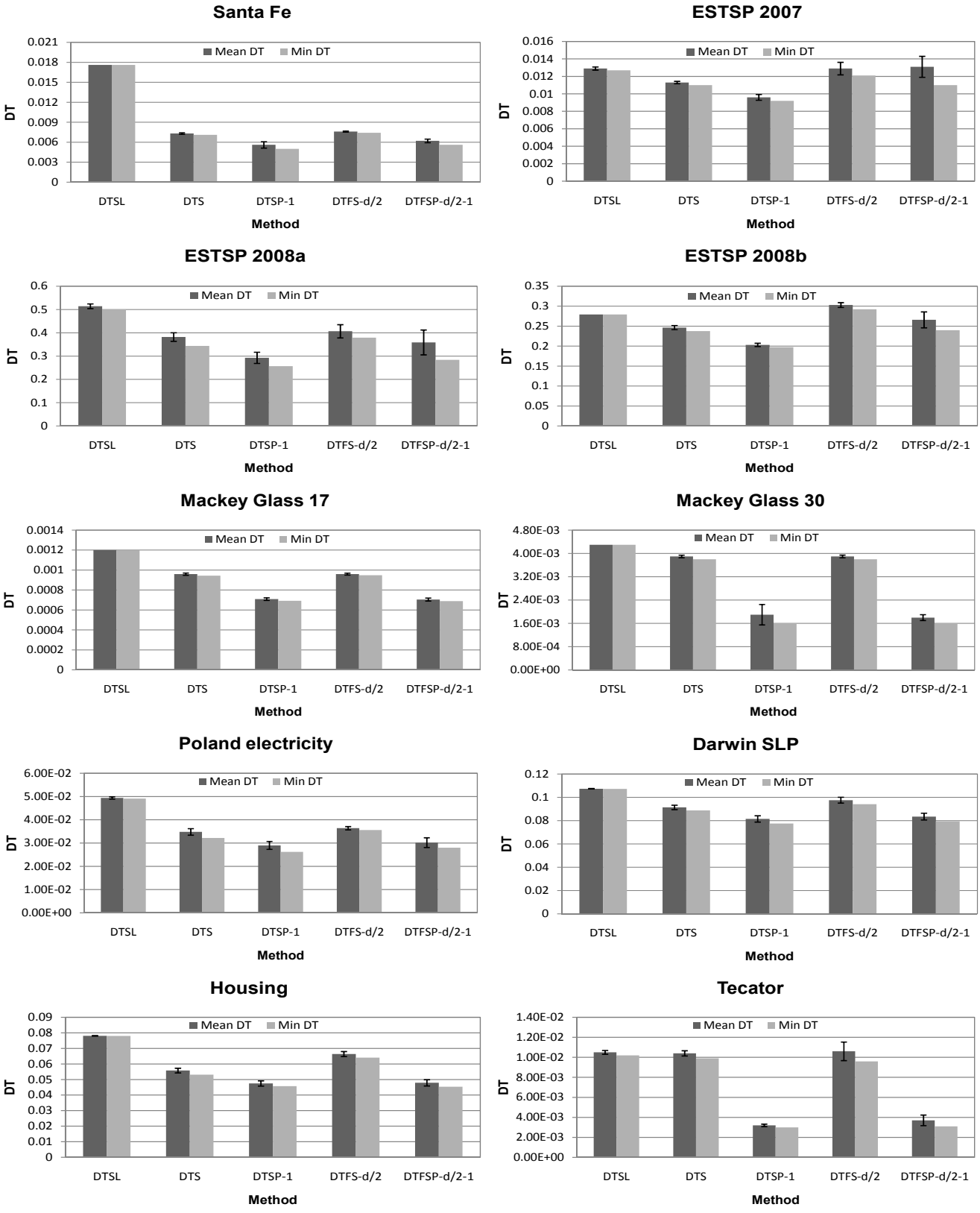


Figure 4: DT performance (average and minimum values) for ten datasets ($\epsilon = 1$).

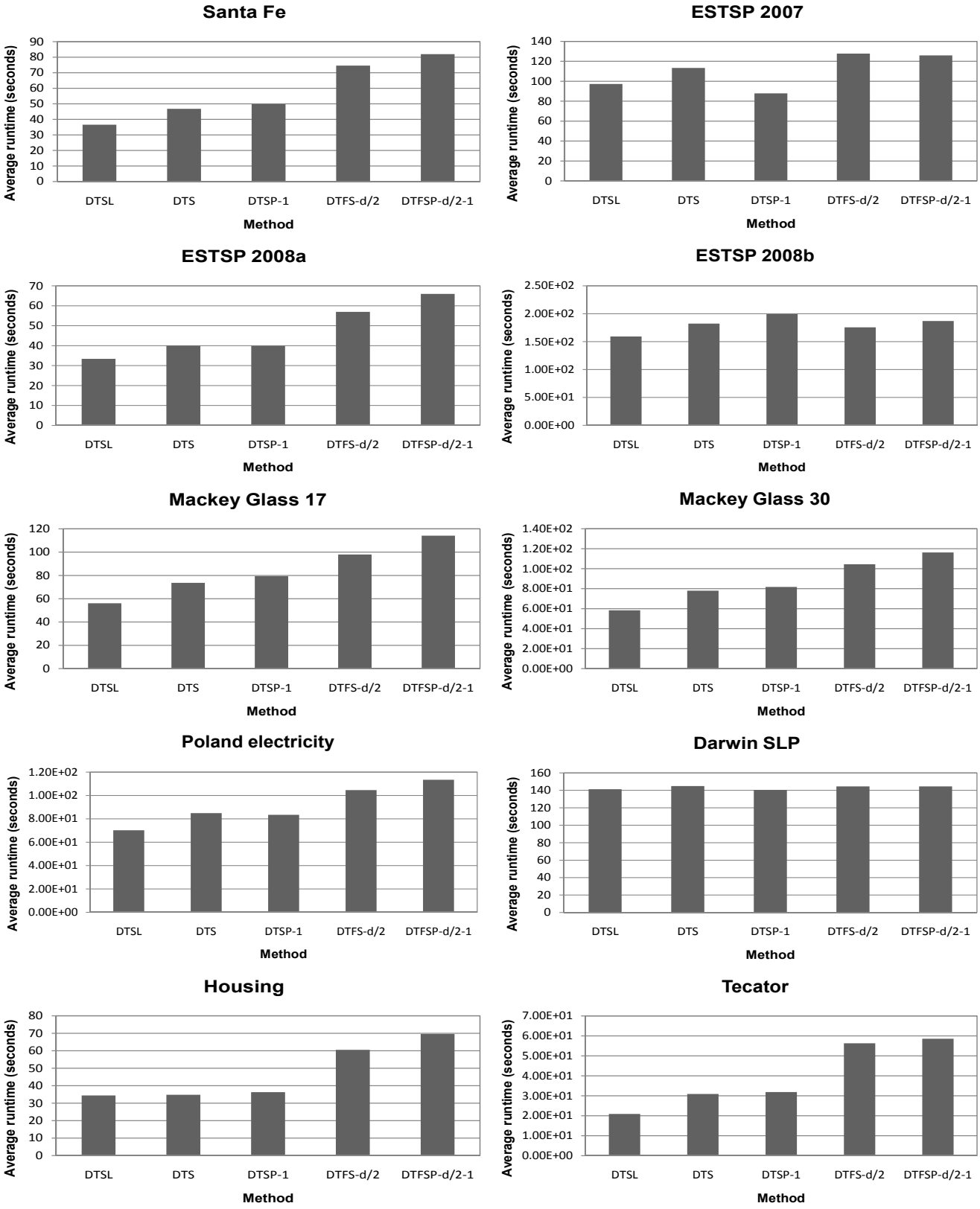


Figure 5: Computational times obtained for ten datasets ($\epsilon = 1$).

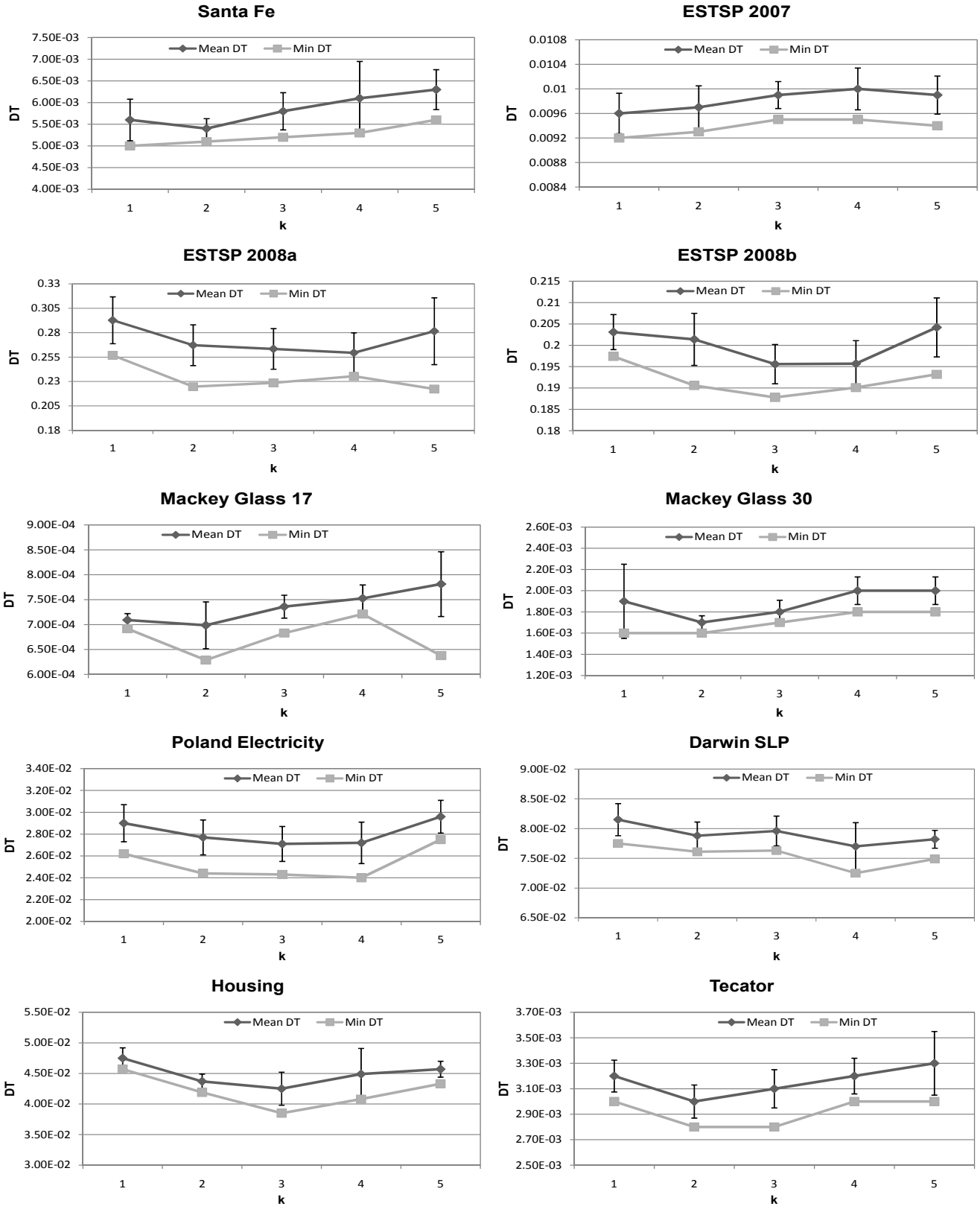


Figure 6: DTSP-k results using projection to $k = \{1, 2, 3, 4, 5\}$ dimensions for ten datasets.

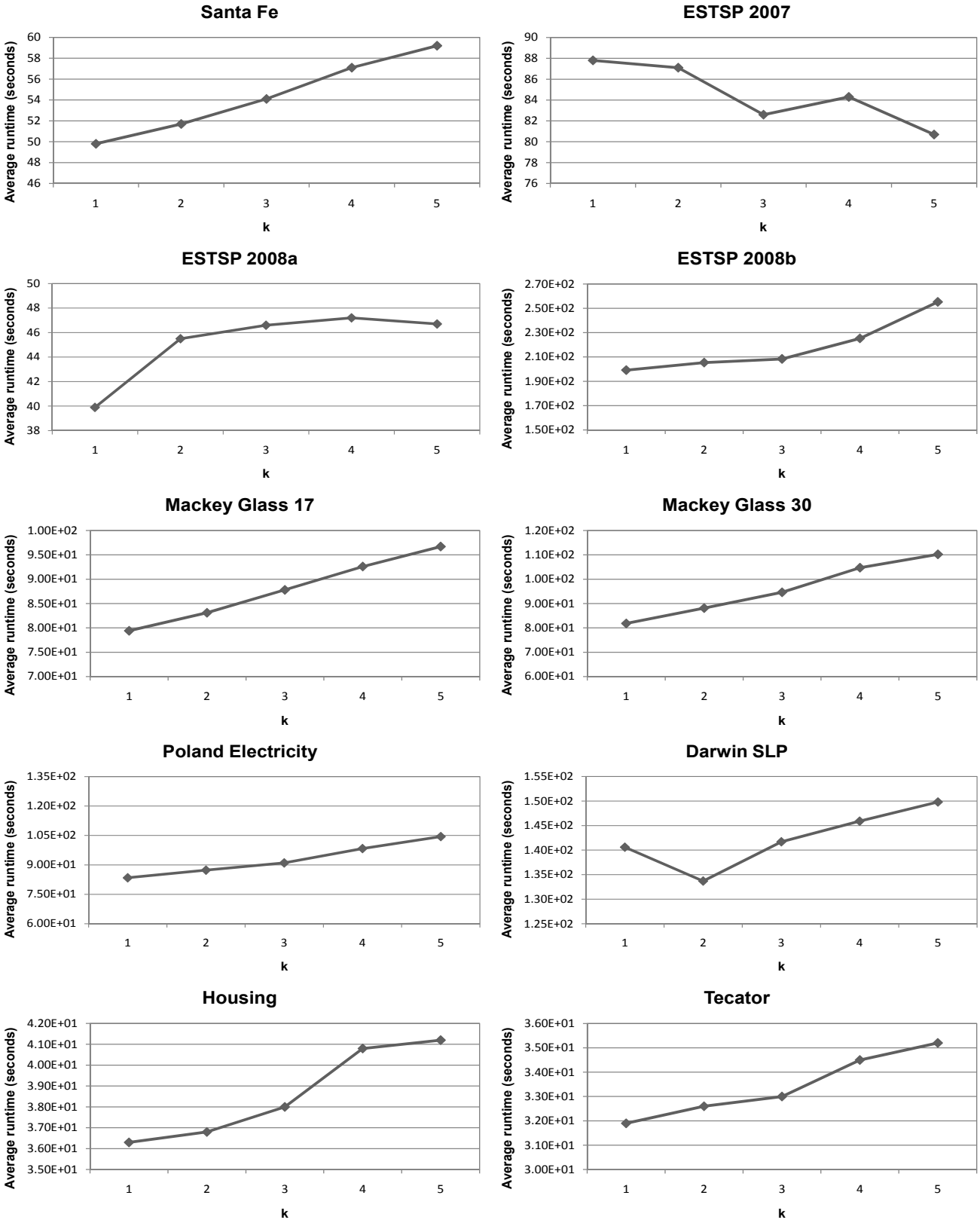


Figure 7: Computational times obtained for DTSP-$\{1, 2, 3, 4, 5\}$ for ten datasets.