Minimizing the Delta Test for Variable Selection in Regression Problems

A. Guillén*
Department of Computer Architecture and Technology,
University of Granada, Spain
E-mail: aguillen@atc.ugr.es
*Corresponding author

D. Sovilj
Department of Information and Computer Science,
Helsinki University of Technology, Finland
E-mail: dusans@cis.hut.fi

F. Mateo
Inst. of Applications of Information Technology and Advanced Communications,
Polytechnic University of Valencia, Spain
E-mail: fermaji@upvnet.upv.es

I. Rojas and A. Lendasse
Department of Computer Architecture and Technology,
University of Granada, Spain
E-mail: irojas@atc.ugr.es
Department of Information and Computer Science,
Helsinki University of Technology, Finland
E-mail: lendasse@hut.fi

Abstract: The problem of selecting an adequate set of variables from a given data set of a sampled function, becomes crucial by the time of designing the model that will approximate it. Several approaches have been presented in the literature although recent studies showed how the Delta Test is a powerful tool to determine if a subset of variables is correct. This paper presents new methodologies based on the Delta Test such as Tabu Search, Genetic Algorithms and the hybridization of them, to determine a subset of variables which is representative of a function. The paper considers as well the scaling problem where a relevance value is assigned to each variable. The new algorithms were adapted to be run in parallel architectures so better performances could be obtained in a small amount of time, presenting great robustness and scalability.

Keywords: Variable selection; delta test; forward-backward search; tabu search; genetic algorithms; hybrid algorithms; parallel architectures.

Reference to this paper should be made as follows: Guillén, A., Sovilj, D., Mateo, F., Rojas, I. and Lendasse, A. (200?) 'New methodologies based on delta test for variable selection in regression problems', Int. J. High Performance Systems Architecture, Vol. ?, No. ?, pp.?–?.

Biographical notes: Fernando Mateo is a MSc researcher pursuing his PhD at the Institute of Applications of Information Technology and Advanced Communications, in the Polytechnic University of Valencia, Valencia, Spain. He obtained his M.Sc. from the Polytechnic University of Valencia in 2005. Since then, he has been researching in machine learning methods and their applications, namely: Position estimation in PET detectors, prediction of contaminants in food and variable selection techniques for high-dimensional problems.

1 INTRODUCTION

In many real-life problems it is convenient to reduce the number of involved features (variables) in order to reduce the complexity, especially when the number of features is large compared to the number of observations (e.g. finance problems, weather forecast, electricity load prediction, medical or chemometric information, etc.). There are several criteria to tackle this variable reduction problem. Three of the most common are: maximization of the mutual information (MI) between the inputs and the outputs, minimization of the k-nearest neighbors (k-NN) leave-one-out generalization error estimate and minimization of a nonparametric noise estimator (NNE).

The problem of regression or function approximation consists in, given a set of input vectors with their corresponding output, it is desired to build a model that learns the relationship between the input variables and the output variable. The designed model should also have the ability to generalize, so when new input vectors that were not in the original training data set are given to the model, it is able to generate the correct output for those values. Formally this problem can be enunciated as, given a set of observations $\{(\vec{x}_j; y_j); j = 1, ..., N\}$ with $y_j = F(\vec{x}_j) \in \mathbb{R}$ and $\vec{x}_j \in \mathbb{R}^d$, it is desired to obtain a function $\mathcal{G}$ so $y_j = \mathcal{G}$ $(\vec{x}_j) \in \mathbb{R}$ with $\vec{x}_j \in \mathbb{R}^d$.

There exists a wide variety of models that are able to approximate any function such as Radial Basis Function Neural Networks (Poggio and Girosi, 1989), Multilayer Perceptrons (Rosenblatt, 1958), Fuzzy Systems (Kosko, 1994), Gaussian Process (Rasmussen, 2004), Support Vector Machines (SVM) and Least Square SVM (Lee and Verri, 2002), etc. however, they all suffer from the Curse of Dimensionality (Herrera et al., 2006). As the number of dimensions d grows, the number of input values required to sample the solution space increases exponentially, this means that the models will not be able to set their parameters correctly if there are not enough input vectors in the training set. Many real life problems present this drawback since they have a considerable amount of variables to be selected in comparison to the few number of observations. Thus, efficient and effective algorithms to reduce the dimensionality of the data sets are required. Another aspect that is improved by selecting a subset of variables is the interpretability of the designed systems (Guyon et al., 2006).

The literature presents a wide number of methodologies for feature or variable selection ((Punch et al., 1993; Oh et al., 2004, 2002; Raymer et al., 2000; Saeys et al., 2007)) although they have been focused on classification problems. Regression problems differ from classification since:

- the output of the problem is continuous, not like in classification, where a finite number of classes is defined a priori. For example, the output of the function could be in the interval [0,1] meanwhile a classification could consist in the distinction between $(table, chair, library)$.
- There is a proximity between different outputs, not like in classification, where classes (in general) cannot be related. For example, it is possible to determine that the output value 0.45 is closer to 0.5 than the value 0.35 but it is not possible to determine a proximity between the class *table* and the class *chair* or *library*.

Therefore, specific algorithms for this kind of problem must be designed. Recently, it has been demonstrated in (Eirola et al., 2008) how the Delta Test (DT) is a quite powerful tool to determine the quality of a subset of variables. The latest work related to feature selection using the DT consisted in the employment of a local search technique such as Forward-Backward. However, there are other alternatives that allow to perform a global optimization of the variable selection like Genetic Algorithms (GA) (Holland, 1975) and Tabu Search (TS) (Glover and Laguna, 1997). One of the main drawbacks of using global optimization techniques is their computational cost. Nevertheless, the latest advances in computer architecture provide powerful clusters without requiring a large budget, so an adequate parallelization of these techniques might ameliorate this problem. This is quite important in real life applications where the response time of the algorithm must be acceptable from the perspective of a human operator. This paper presents several new approaches to perform variable selection using the DT as criterion to decide if a subset of variables is adequate or not. The new approaches are based in local search methodologies, global optimization techniques and the hybridization of both. The rest of the paper is structured as follows: Section 2 introduces the DT and its theoretical framework, then, Section 3 describes the previous methodology to perform the variable selection as well as the new developed algorithms. Section 4 presents a complete experimental study analyzing the behavior of the algorithms and, in Section 5, conclusions are drawn.

2 THE DELTA TEST

The Delta Test (DT), firstly introduced by Pi and Peterson for time series (Pi and Peterson, 1994) and proposed for variable selection in (Eirola et al., 2008), is a technique to estimate the variance of the noise, or the mean squared error (MSE), that can be achieved without overfitting. Given N input-output pairs $(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$, the relationship between $\mathbf{x}_i$ and y_i can be expressed as

$$y_i = f(\mathbf{x}_i) + r_i, \qquad i = 1, ..., N$$

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

The DT is useful for evaluating the nonlinear correlation between two random variables, namely, input and output pairs. The DT can be also applied to input variable selection: the set of input variables that minimizes the DT is

the one that is selected. Indeed, according to the DT, the selected set of input variables is the one that represents the relationship between input variables and the output variable in the most deterministic way. DT is based on hypothesis coming from the continuity of the regression function. If two points $\mathbf{x}$ and $\mathbf{x}'$ are close in the input variable space, the continuity of regression function implies the outputs $f(\mathbf{x})$ and $f(\mathbf{x}')$ will be close enough in the output space. Alternatively, if the corresponding output values are not close in the output space, this is due to the influence of the noise.

The DT can be interpreted as a particularization of the Gamma Test (Jones, 2004) considering only the first nearest neighbor. Let us denote the first nearest neighbor of a point $\mathbf{x}_i$ in the $\mathbb{R}^d$ space as $\mathbf{x}_{NN(i)}$. The nearest neighbor formulation of the DT estimates $\mathrm{Var}[r]$ by

$$\mathrm{Var}[r] \approx \delta = \frac{1}{2N}\sum_{i=1}^{N}(y_i - y_{NN(i)})^2,$$
$$\text{with } \mathrm{Var}[\delta] \to 0 \text{ for } N \to \infty$$

where $y_{NN(i)}$ is the output of $\mathbf{x}_{NN(i)}$.

3 VARIABLE SELECTION METHODOLOGIES

This Section presents the previous methodology proposed to compute the minimum value for the DT, the Forward-Backward search. Then, it presents an adaptation of the Tabu search for the minimization of the DT. Finally, a new algorithm that combines the Tabu Search and the global optimization capabilities of GA is introduced.

The methodologies described below are applied to the problem of variable selection from two perspectives: binary and scaled. The binary approach considers a solution as a set of binary values that indicate if that variable is relevant or not. The scaled approach assigns a weighting factor to each variable according to its importance. The scaling problem is more challenging because the solution space grows considerably.

3.1 Forward-backward search

To overcome the difficulties and the high computational time that an exhaustive search would entail (i.e. $2^d - 1$ input variable combinations, being d the number of variables), there are several other search strategies. These strategies are affected by local optima because they do not test every input variable combination, but they are preferred over an exhaustive search if the number of variables is too large.

Among the typical search strategies, there are three that share similarities:

- Forward search
- Backward search (or pruning)
- Forward-backward search

The difference between the first two is that the forward search starts from an empty set of selected variables and adds variables to it according to the optimization of a search criterion, while the backward search starts from a set containing all the variables and removes those for which the elimination optimizes the search criterion.

Both forward and backward search suffer from incomplete search. The Forward-Backward Search (FBS) is a combination of them. It is more flexible in the sense that a variable is able to return to the selected set once it has been dropped, and vice versa, a previously selected variable can be discarded later. This method can start from any initial input variable set: empty set, full set, custom set or randomly initialized set. If we consider a set of N input-output pairs $(x_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$, the forward-backward search algorithm can be described as follows

1. Initialization:

   Let S be the selected input variable set, which can contain any input variables, and F the unselected input variable set, which contains the variables not present in S. Compute $\mathrm{Var}[r]$ using Delta Test on the set S.

2. Forward-Backward selection step:

   Find the variable x^S to include or to remove from the set S to minimize $\mathrm{Var}[r]$:

   $$x^S = \arg\, min_{x^i, x^j}\{(\mathrm{Var}[r]|S \cup x^j) \cup (\mathrm{Var}[r]|S \setminus x^i)\},$$
   $$x^i \in S, x^j \in F$$

3. If the old value of $\mathrm{Var}[r]$ on the set S is lower than the new result, stop; otherwise, update set S and save the new $\mathrm{Var}[r]$. Repeat step 2 until S is equal to any former selected S.

4. The selected input variable set is S

3.2 Tabu search

Tabu Search (TS) is a metaheuristic method designed to guide local search methods to explore the solution space beyond local optimality. The first most successful usage was by Glover (Glover, 1986, 1989, 1990) for combinatorial optimization. Later TS was successfully used in scheduling (DellÁmico and Trubian, 1993; Mantawy et al., 2002; Zhang et al., 2007), design (Xu et al., 1996a,b), routing (Brandao, 2004; Scheuerer, 2006) and general optimization problems (Glover, 2006; Hedar and Fukushima, 2006; Al-Sultan and Al-Fawzan, 1997). The TS has become a powerful method with different components tied together, that is able to obtain excellent results in different problem domains.

In general, the problem is in form of an objective or cost function $f(v)$, given the set of solutions $v \in V$. In the context of TS, the neighborhood relationship between solutions, denoted $Ne(v)$, plays the central role. While

there are other neighborhood based methods, such as descent/ascent methods widely used, the difference is that tabu uses memory in order to influence which parts of the neighborhood are going to be explored. A memory is used to record various aspects of the search process and the solutions encountered, such as recency, frequency, quality and influence of moves. Instead of storing whole solutions in memory, which is impractical in some problems, the common thing is to store attributes of solutions or moves/operations used to transition from one solution to the next one.

The most important aspect of the memory is to forbid some moves to be applied, or in other words, to prevent the search to go back to solutions that were already visited. This also allows the search to focus on such moves that guide the search toward unexplored areas of the solutions space. This part of the memory is called a *tabu list*, and the moves in this list are then considered tabu, and thus forbidden to use. The size of the tabu list as well as the time each move is kept in the list are important issues in TS. These parameters should be set up so that the search is able to go through two distinct, but equally important phases: *intensification* and *diversification*.

As far as we know, no implementation of the TS to minimize the DT has ever been done, so it was implemented in this work as an improvement over the FBS, which does not have any memory enhancement. Due to the fact that this paper considers the variable selection and the scaling problem, two different algorithms had to be designed. Both algorithms use only short-term recency based memory to store *reverse moves* instead of solutions to speedup the exploration of the search space.

3.2.1 TS for pure variable selection

In the case of variable selection, a move was defined as a flip of the status of exactly one variable in the data set. The status is excluded (0) or included (1) from the selection. For a data set of dimensionality d, a solution is then a vector of zeros and ones $v = (v_1, v_2, \ldots, v_d)$, where $v_k \in \{0, 1\}, k = 1, ..., d$, are indicator variables representing the selection status of k-th dimension.

The neighborhood of a selection (solution) v is a set of selections u which have exactly one variable that has different status. This can be written as

$$Ne(v) = \{u \mid \exists_1 q \in \{1, \ldots, d\} \ v_q \neq u_q \wedge v_i = u_i, i \neq q\}$$

With this setup, each solution has exactly the same amount of neighbors, which will be equal to d.

3.2.2 TS for the scaling problem

The first modification that requires the adaptation to the scaling problem is the definition of the neighborhood of a solution. A solution v is now a vector with scaling values from a discretized set $v_k \in H = \{0, 1/k, 2/k, \ldots, 1\}$, where k is discretization parameter. Two solutions are neighbors if they *disagree* on exactly one variable, same as for variable selection, but the disagreement is the smallest possible value. $Ne(v)$ is defined in a same way as for variable selection, but with an additional constraint of $|v_q - u_q| = 1/k$. For example, for $k = 10$ and $d = 3$, the solutions $v_1 = (0.4, 0.2, 0.8)$ and $v_2 = (0.3, 0.2, 0.8)$ would be neighbors, but not the solution $v_3 = (0.1, 0.2, 0.8)$. The move between solutions is defined as a change of value for one dimension, which can be written as a vector (dimension, old value, new value).

3.2.3 Setting the tabu conditions

The *tenure* for a move is defined as the number of iterations that it is considered as tabu. This value is determined empirically when the TS is applied to solve a concrete problem. For the variable selection problem, this paper proposes a value which is dependent on the number of dimensions so it can be applied to several problems. In the experiments, two tabu lists, and thus two tenures, were used. The first list is responsible for preventing the change along certain dimension for $d/4$ iterations. The second one prevents the change along the same dimension and for specified scaling value for $d/4 + 2$ iterations. The combination of these two lists gave better results than when each of the conditions was used alone.

For example, for $k = 10$, if a move is performed along dimension 3 from value 0.1 to 0.2, which can be written as a vector $m = (3; 0.1, 0.2)$, then its reverse move $m^{-1} = (3; 0.2, 0.1)$ is stored in the list. The search will be forbidden to use any move along dimension 3 for $d/4 + 2$ iterations, and after that time, it will be further 2 iterations restricted to use the move m^{-1}, or in other words to go back from 0.2 to 0.1.

With these settings, in the case of variable selection, two conditions are then implicitly merged into one condition: restrict a flip of the variable for $d/4 + 2$ iterations. This is because there are only two values 0,1 as possible choices.

3.3 Hybrid Parallel Genetic Algorithm

The benefits and advantages of the global optimization and local search techniques have been hybridized in the proposed algorithm. The idea is to be able to have a global optimization, using a GA, but still being able to make a fine tune of the solution, using the TS. The following paragraphs describe the different elements that define the algorithm.

3.3.1 Encoding of the individuals and initial population

Deciding how a chromosome encodes a solution is one of the most decisive design steps since it will influence the rest of the design (De Jong, 1996; Mitchell and Forrest, 1995; Michalewicz, 1996). The classical encoding used for variable selection has been a vector of binary values where 1 represents that the variable is selected and 0 that the variable is not selected. As it has been commented above,

this paper is considering the variable selection using scaling in order to determine the importance of a variable. Therefore, instead of using binary values, other encoding must be chosen. If instead of using 0 and 1, the algorithm uses real numbers to determine the weight of a variable, the GA could fall into the category of Real Coded Genetic Algorithms (RCGA). However, the number of scales has been discretized in order to bound the number of possible solutions making the algorithm a classical GA where the cardinality of the alphabet for the solutions is increased in k values. For the sake of simplicity in the implementation, an individual is a vector of integers where the number 1 represents that the variable was selected and $k+1$ means that the variable is not selected.

Regarding the initial population, some individuals are included in the population deterministically to ensure that each scaling value for each variable exists in the population. These individuals are required if the classical GA crossover operators (one/two-points, uniform) are applied so all the possible combinations can be reached. For example, if the number of scales is 4 in a problem with 3 variables, the individuals that are always included in the population are: 1 1 1, 2 2 2, 3 3 3, and 4 4 4.

3.3.2 Selection, crossover and mutation operators

The algorithm was designed in order to be as fast as possible so when several design options appeared, the fastest one (in terms of computation time) was selected, as long as it was reasonable. The selection operator chosen was the binary tournament selection as presented by Goldberg in (Goldberg, 1989) instead of the Baker's roulette wheel operator (Baker, 1987) or other more complex operators existing in the literature (Chakraborty et al., 1996). The reason for this is because the binary tournament does not require the computation of any probability for each individual, thus, a considerable amount of operations are saved on each iteration. This is specially important for large populations. Furthermore, the fact that the binary tournament does not introduce a high selective pressure is not as traumatic as it might seem. The reason is because the huge solution space, that arises as soon as the number of scales increases, should be deeply explored to avoid local minima. Nevertheless, the algorithm incorporates the elitism mechanism, keeping the 10% of the best individuals of the population, so the convergence is still feasible.

Regarding the crossover operator, the algorithm implemented the classical operators for a binary coded GA, these are: one-point and two-point crossovers and the uniform crossover (De Jong, 1975; Holland, 1975; Sywerda, 1989). The behavior of the algorithm using these crossovers was quite similar and acceptable. Nonetheless, since the algorithm could be included into the Real Coded GA class, an adaptation of the BLX-α (Eshelman and Schaffer, 1993) was implemented as well. The operator 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 an 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 adaptation only required to round the absolute value assigned to each gene and also the modification of the value in case it is out of the bounds of the solution space.

The mutation operates at a gene level, so a gene has the chance to get any value of the alphabet.

3.3.3 Parallelization

The algorithm has been parallelized so it is able to take advantage of the parallel architectures, like clusters of computers, that are easy available anywhere. The main reason to parallelize the algorithm is to be able to explore more solutions in the same time, allowing the population to reach a better solution. There is a wide variety of possibilities when parallelizing a GA (Cantu-Paz, 2000; Alba and Tomassini, 2002; Alba et al., 2004), however, as a first approach, the classical master/slave topology has been chosen (Grefenstette, 1981).

As the previous subsection commented, the algorithm was designed to be as fast as possible, nonetheless, the fitness function still remains expensive in comparison with the other stages of the algorithm (selection, crossover, mutation, etc.). At first, all the stages of the GA were parallelized, but the results showed that the communication and synchronization operations could be more expensive than performing the stages synchronously and separately on each processor [1]. Hence, only the computation of the DT for each individual is distributed between the different processors. As it is proposed in the literature (Cantu-Paz, 2000), the master/slave topology uses one processor to perform the sequential part of the GA and then, sends the individuals to different processors that will compute the fitness function. Some questions might arise at this point like: Are the processors homogeneous?, How many individuals are sent at a time?, Is the fitness computation time constant?

The algorithm assumes that all processors are equal with the same amount of memory and speed. If they were not, it should be considered to send the individuals iteratively to each processors as soon as they were finished with the computation of the fitness of an individual. This is equivalent to the case where the fitness function computational time might change from one individual to another. However, the computation of the DT does not significantly vary from one individual to another, despite the number of variables they have selected. Thus, using homogeneous processors and constant time consuming fitness function, the amount of individuals that each processor should evaluate is *size of population/number of processors*.

The algorithm has been implemented so the number of communications (and their number of packets) is minimized. To achieve this, all the processors execute exactly

[1] This paper considers that a processor will execute one process of the algorithm.

the same code so, when each one of them has to evaluate its part of the population, it does not require to get the data from the master because it already has the current population. The only communications that have to be done during the execution of the GA are, after the evaluation of the individuals, to send and receive the values of the DT, but not the individuals themselves. To make possible that all processors have the same population all the time considering that there are random elements, at the beginning of the algorithm the master processor sends to all the others the seed for their random number generators. This implies that they all produce the same values when calling the function to obtain a random number. In this way, when the processors have to communicate, only the value of the DT computed will be sent, saving the communications that would require to send each individual to be evaluated. This is specially important since some problems require large individuals, increasing the traffic in the network and retarding the execution of the algorithm.

3.3.4 Hybridization

The combination of local optimization techniques with GA has been studied in several papers (Ishibuchi et al., 2003; Deb and Goel, 2001; Guillén et al., 2008). Furthermore, the inclusion of a FBS in a stage of an algorithm for variable selection was recently proposed in (Oh et al., 2004) but, the algorithm was oriented to classification problems. The new approach that this paper proposes is to perform a local search at the beginning and at the end of the GA. The local search will be done by using the TS described in the Section 3.2, so it does not stop when it finds a local minimum. Using a good initialization as a start point for the GA, it is possible to find good solutions with smaller populations (Reeves, 1993). This is quite important because the smaller the population is, the faster the algorithm will complete a generation. Therefore, the algorithm incorporates an individual generated using the TS, so there is a potential solution which has a good fitness. Since the algorithm i s able to use several processors, several TSs can be run on each processor. The processors will communicate sending each other the final individual once the TS is over. Afterwards, they will start the GA using the same random seed. Thus, if there are p processors, p individuals of the population will be generated using the TS. Thanks to the use of the binary tournament selection, there is no need to worry about converging too fast to a local minimum near these individuals. The GA will then explore the solution space and when it finishes, each processor will take an individual using it as the starting point for a new TS. In this way, the exploitation of a good solution is guaranteed. The first processor takes the best individual, the second, takes the second best individual and so on. As the GA maintains diversity in the population, the best result after applying the TS does not always come from the best individual in the population. This fact shows how important is to ke ep exploring the solution space instead of letting the GA converge too fast.

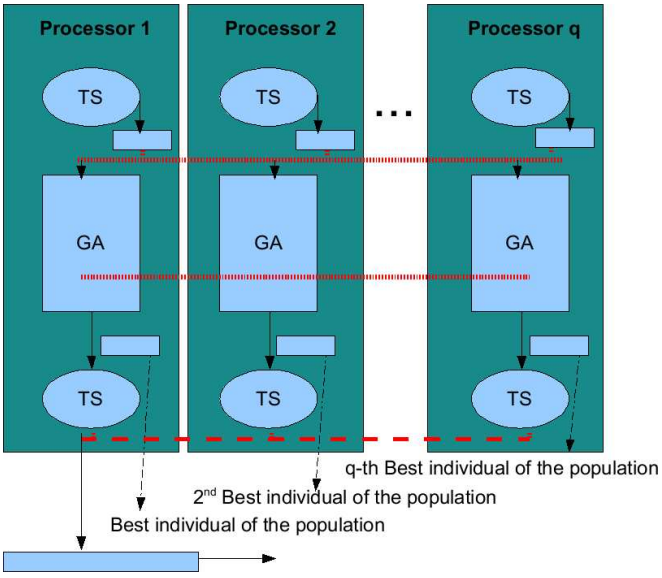

Figure 1: Algorithm scheme. Dashed line represents one to one communication, dotted lines represent collective communications.

Figure 1 shows how the algorithm is structured as well as the communications that are necessary to obtain the final solution.

4 EXPERIMENTS AND RESULTS

This Section will show empirically that all the elements described in the previous Section, when they are combined together, could improve the performance. First, the data sets that have been used are introduced. Then, the effect of the parallelism applied over the GA will be analyzed. Afterwards, more experiments with the parallel version will be done in order to show how the addition of the BLX-α crossover and the TS can improve the results. Finally, the new proposed algorithm will be compared against the local search technique used so far, demonstrating how the global optimization, in combination with the local search, leads to better solutions. Nothing was commented so far about the stopping criterion of the algorithm. In the experiments, a time limit of 600 seconds for all the algorithms was used. The decision to set this time limit is because the experience when working with industries says that 10 minutes is the maximum amount of time that an operator is willing to wait. Furthermore, this value has been widely used in the literature as time limit (Wang and Kazmierski, 2005; Zhang et al., 2006). Two different clusters were used in the experiments but, due to the lack of space, only the results of the better one will be showed. Nonetheless, the algorithms had a similar performance in both of them. A remarkable fact was that the size of the cache was crucial by the time of computing the DT. Due to the large size of the distances matrix, the faster computer had a worse performance because it did not have as much cache memory

as the other. Further research must be done regarding the distribution of the samples between the processors so less memory accesses have to be done. The processors in the cluster used had the following characteristics:

Cpu family : 6
Model : 15
Model name : Intel(R) Xeon(R) CPU E5320 @ 1.86GHz
Stepping : 7
Cpu MHz : 1595.931
Cache size : 4096 KB
Cpu cores : 2
Bogomips : 3723.87
Clflush size : 64
Cache alignment : 64
Address sizes : 40 bits physical, 48 bits virtual

The algorithms were implemented in MATLAB and, in order to communicate the different processes, the MPImex ToolBox presented in (Guillen et al., 2008) was used.

4.1 Data sets used in the experiments

To assess the presented methods, several experiments were performed, using the following data sets:

1. The Housing data set[2]: The housing data set is related to the estimation of housing values in suburbs of Boston. The value to predict is the median value of owner-occupied homes in $1000's. The data set contains 506 instances, with 13 input variables and one output.

2. The Tecator data set[3]: The Tecator data set aims at performing the task of predicting the fat content of a meat sample on the basis of its near infrared absorbance spectrum. The data set contains 215 useful instances for interpolation problems, with 100 input channels, 22 principal components (which will remain unused) and 3 outputs, although only one is going to be used (fat content).

3. The Anthrokids data set[4]: This data set represents the results of a three-year study on 3900 infants and children representative of the U.S. population of year 1977, ranging in age from newborn to 12 years of age. The data set comprises 121 variables and the target variable to predict is children's weight. As this data set presented many missing values, a prior sample and variable discrimination had to be performed to build a robust and reliable data set. The final set[5] without missing values contains 1019 instances, 53 input variables and one output (weight). More information on this data set reduction methodology can be found in (Mateo and Lendasse, 2008).

4. The Finance data set[5]: This data set contains information of 200 French industries during a period of 5 years. The number of samples is 650. It contains 35 input variables, related to balance sheet, income statement and market data, and one output variable, called "return on assets" (ROA). This is an indicator of how profitable a company is relative to its total assets. It is usually calculated by dividing a company's annual earnings by its total assets.

5. The Santa Fe time series competition data set[6]: The Santa Fe data set is a time series recorded from laboratory measurements of a Far-Infrared-Laser in a chaotic state, and proposed for a time series competition in 1994. The set contains 1000 samples, and it was reshaped for its application to time series prediction using regressors of 12 samples. Thus, the set used in this work contains 987 instances, 12 inputs and one output.

6. The ESTSP 2007 competition data set[5]: This time series was proposed for the European Symposium on Time Series Prediction 2007. It is an univariate set containing 875 samples but has been reshaped using a regressor of 55 variables, producing a final set of 819 samples, 55 variables and one output.

All the data sets were normalized to zero mean and unit variance, so the DT values obtained are normalized by the variance of the output.

4.2 Parallelization of the GA

This subsection will show the benefits that are obtained by adding parallel programming to the sequential GA. The sequential version was designed exactly in the same way that was described in Section 3, using the same operators, however, the evaluation of the individuals was performed uniquely in one processor. For these experiments, the TS was not incorporated to the algorithms so the benefits of the parallelism could be more easily appreciated.

For these initial tests, the GA parameters were adjusted to the following values:

- Crossover Type: One point crossover
- Crossover Rate: 0.85
- Mutation Rate: 0.1 [7].
- Generational elitism: 10%

The results were obtained from three of the data sets, namely Anthrokids, Tecator and ESTSP competition data set. The performances are presented in Table 1, including a statistical analysis of the values of DT and the number of generations evaluated. Figures 2, 3, 4, 5, 6 and

[2] http://archive.ics.uci.edu/ml/data sets/Housing.
[3] http://lib.stat.cmu.edu/data sets/tecator.
[4] http://ovrt.nist.gov/projects/anthrokids.
[5] http://www.cis.hut.fi/projects/tsp/index.php?page=timeseries.
[6] http://www-psych.stanford.edu/~andreas/Time-Series/SantaFe.html
[7] This is the same rate used in (Oh et al., 2004), also for a feature selection application

7 show the effect of increasing the number of processors in the number of generations done by the algorithms for a constant number of individuals. As it was expected, if the number of individuals increases, the number of generations is smaller. This effect is compensated with the introduction of more processors that increase almost linearly the number of generations completed. The linearity is not that clear for small populations with 50 individuals since the communication overheads start to be significant. However, larger population sizes guarantee a quite good scalability for the algorithm.

Once the superiority of the parallel approach was proved, the next sections will only consider the parallel implementations of the GA.

4.3 Hybridization of the pGA with the TS and using the BLX-α crossover

Once it has been demonstrated how important is the parallelization of the algorithm, the benefits of adapting the BLX-α operator to the parallel version used in the previous subsection will be shown. Furthermore, the comparison will also consider the hybridization with the TS at the beginning and at the end of the algorithm to make a fine tune of the solution provided by the GA. This hybrid algorithm has received the name of pTBGA. The time limit of 600 seconds was divided into three time slices that were assigned to the three different stages of the algorithm: tabu initialization, GA, and tabu refinament.

The goal was to find the best trade-off in terms of time dedicated to explore the solution space and to exploit it. When assigning the time slices to each part it was considered that, for the initialization using the TS, just a few evaluations were willed in order to improve slightly the starting point for the GA. Several combinations where tried but always keeping the time for the first TS smaller than the GA and the second TS.

The results are listed in Table 2. The Tables present a comparison between pRCGA, pTBGA using only TS at the end, and pTBGA using TS at the beginning and at the end, both with and without scaling. The population size was fixed to 150 individuals based on previous experiments where no significant difference was observed over 200 individuals if the number of processors is fixed to 8. The configuration of pTBGA is indicated in Table 2 as $t_{TS_1}/t_{GA}/t_{TS_2}$ where t_{GA} is the time (in seconds) dedicated to the GA, t_{TS_1} is the time dedicated to the first tabu step, and t_{TS_2} the time dedicated to the last.

The values of DT obtained show how the application of the adapted BLX-α crossover improves the results for pRCGA. Regarding the hybridization, there is no doubt that introducing the TS to the algorithm improves the results significantly. The effect of introducing the TS before the start of the GA improves the results in some cases, although the improvement is not too significant. However, it is possible to appreciate how the application of the TS at the beginning and at the end reduces the standard deviation making the algorithm more robust.

4.4 Comparison against the classical methodologies

This last Subsection performs a comparison of the final version of the proposed algorithm in this paper with the classical local search methodology already proposed in the literature to minimize the value of the DT. The pTBGA with optimal settings running on 8 processors was compared in terms of performance (minimum Delta Test and number of solutions evaluated) with other widely used sequential search methods such as FBS and the TS presented in this paper, both running on single processors of the grid. As FBS converged rather quickly (always before the time limit of 10 minutes), the algorithm was run with several initializations, until the time limit was reached. The results of these tests appear listed in Table 3.

For the pTBGA, a fixed population of 150 individuals was selected. The crossover probability was 0.85 in all cases.

When comparing the two local search techniques, this is, TS and FBS, it is remarkable the good behavior of FBS against the TS. This is not surprising since the FBS, as soon as it converged, it was reinitialized starting from another random position. On the other hand, the TS started at one random point and explored the neighborhood of it during the time frame specified, making it more difficult to explore other areas. The new hybrid approach improves the results of the FBS in average for both pure selection and scaling, being more robust than the FBS which does not always provide a good result.

5 CONCLUSIONS

This paper has presented a new approach to solve the problem of simple and scaled variable selection. The major contributions of the paper are:

- The development of a TS algorithm for both, pure selection and scaling, based on the Delta Test. A first initialization of the parameters required by the short time memory was proposed as well
- The design of a Genetic Algorithm whose fitness function is the Delta Test and that makes a successful adaptation of the BLX-α crossover to adapt the discretized scaling problem as well as the pure variable selection.
- The parallel hybridization of the two previous algorithms that allows to keep the compromise between the exploration/exploitation allowing the algorithm to find smaller values for the Delta Test than the previous methodology does.

The results showed how the synergy of different paradigms can lead to obtain better results. It is also important to notice how necessary is the addition of parallelism in the methodologies since the increasing size of the data sets will

Table 1: Performance of RCGA vs pRCGA for three different data sets. Values of the DT and number of generations completed.

Data set	Population	Measurement	RCGA		pRCGA (np=2)		pRCGA (np=4)		pRCGA (np=8)	
			k=1	k=10	k=1	k=10	k=1	k=10	k=1	k=10
Anthrokids	50	Mean (DT)	0.01278	0.01527	0.01269	0.01425	0.01204	0.01408	0.01347	0.0142
		Mean (Gen.)	35.5	16.7	74.8	35.3	137.8	70	169.3	86
	100	Mean (DT)	0.01351	0.01705	0.01266	0.01449	0.01202	0.0127	0.0111	0.01285
		Mean (Gen.)	17.2	8.5	35.4	17.3	68.8	35	104	44.5
	150	Mean (DT)	0.01475	0.01743	0.01318	0.0151	0.01148	0.01328	0.01105	0.01375
		Mean (Gen.)	11	5.7	22.7	11.2	45.6	23.2	61	31
Tecator	50	Mean (DT)	0.13158	0.14151	0.14297	0.147	0.13976	0.14558	0.1365	0.1525
		Mean (Gen.)	627	298.1	1129.4	569.5	2099.2	1126.6	3369.5	1778.5
	100	Mean (DT)	0.13321	0.14507	0.13587	0.14926	0.13914	0.14542	0.13525	0.1466
		Mean (Gen.)	310.8	154.4	579.6	299.9	1110.4	583	1731	926.5
	150	Mean (DT)	0.13146	0.14089	0.1345	0.15065	0.13522	0.14456	0.1303	0.1404
		Mean (Gen.)	195	98.3	388.1	197.8	741.2	377	1288	634.5
ESTSP	50	Mean (DT)	0.01422	0.01401	0.01452	0.01413	0.01444	0.014	0.01403	0.0142
		Mean (Gen.)	51	29.1	99.2	57.6	190.8	113.8	229	126.7
	100	Mean (DT)	0.01457	0.01445	0.01419	0.01414	0.01406	0.01382	0.01393	0.01393
		Mean (Gen.)	24.8	14	50.5	27.9	93	57.8	128.7	67.7
	150	Mean (DT)	0.01464	0.01467	0.01429	0.01409	0.01402	0.01382	0.0141	0.01325
		Mean (Gen.)	16.6	9.1	33.6	18.7	63.2	37.6	82.5	49.5

Table 2: Performance of pRCGA vs pTBGA, with the BLX-α crossover operator

Data set	Measurement	pRCGA (np=8)		pTBGA (np=8) 0/400/200		pTBGA (np=8) 50/325/225	
		k=1	k=10	k=1	k=10	k=1	k=10
Anthrokids	Mean (DT)	0.0113	0.0116	0.0084	0.0103	0.0083	0.0101
	StDev (DT)	11.5e-4	14.7e-4	17.3e-5	53.1e-5	5.8e-5	83.3e-5
Tecator	Mean (DT)	0.13052	0.1322	0.1180	0.1303	0.1113	0.1309
	StDev (DT)	25.8e-4	27.2e-4	12.1e-3	20.7e-4	88.9e-4	10.6e-4
ESTSP	Mean (DT)	0.01468	0.01408	0.01302	0.01308	0.01303	0.0132
	StDev (DT)	16.4e-5	13e-5	8.4e-5	37.7e-5	5.8e-5	17.3e-5
Housing	Mean (DT)	0.0710	0.0584	0.0710	0.0556	0.0710	0.0563
	StDev (DT)	0	9.2e-4	0	8.5e-4	0	6.2e-4
Santa Fe	Mean(DT)	0.0165	0.0094	0.0165	0.0092	0.0165	0.0092
	StDev (DT)	0	9.8e-5	0	11.5e-5	0	11.5e-5
Finance	Mean(DT)	0.1498	0.1371	0.1406	0.1244	0.1406	0.1235
	StDev (DT)	3.4e-4	3.7e-4	7.9e-4	6.1e-3	1.2e-4	6.2e-4

Table 3: Performance comparison of FBS, TS and the best pTBGA configuration

Data set	Measurement	FBS		TS		pTBGA (np=8)*	
		k=1	k=10	k=1	k=10	k=1	k=10
Anthrokids	Mean (DT)	0.00851	0.01132	0.00881	0.01927	0.00833	0.0101
	StDev (DT)	17.3e-5	19.1e-4	27.3e-5	36.5e-4	5.8e-5	83.3e-5
Tecator	Mean (DT)	0.13507	0.14954	0.12799	0.18873	0.1113	0.1309
	StDev (DT)	37.7e-4	10.6e-3	24.7e-4	10.4e-3	88.9e-4	10.6e-4
ESTSP	Mean (DT)	0.01331	0.01415	0.01296	0.01556	0.01302	0.01308
	StDev (DT)	28.8e-5	44e-5	26.3e-5	16.4e-4	8.4e-5	37.7e-5
Housing	Mean (DT)	0.0710	0.0586	0.0711	0.0602	0.0710	0.0556
	StDev (DT)	0	44.7e-5	35.4e-5	87.3e-4	0	8.5e-4
Santa Fe	Mean (DT)	0.0165	0.00942	0.0178	0.0258	0.0165	0.0092
	StDev (DT)	0	7.1e-5	24.6e-4	18.5e-3	0.0165	0.0091
Finance	Mean (DT)	0.1411	0.1377	0.1420	0.4381	0.1406	0.1235
	StDev (DT)	12.7e-4	46.6e-4	32.8e-4	0.1337	16.2e-4	64.2e-4

* The best pTBGA configuration among the tested for each data set.

not be able to be processed by monoprocessor architectures. Regarding future research, this paper has addressed the problem of the cache memory limitation that seems quite relevant for large data sets. Also, further work on the study of distributed demes genetic algorithms must be done.

ACKNOWLEDGMENT

This work has been partially supported by the projects TIN2007-60587, P07-TIC-02768 and P07-TIC-02906, and by research grant BES-2005-9703 from the Spanish Ministry of Science and Innovation.

REFERENCES

Al-Sultan, K. S. and Al-Fawzan, M. A. (1997). A tabu search hooke and jeeves algorithm for unconstrained optimization. *European Journal of Operational Research*, 103(1):198–208.

Alba, E., Luna, F., and Nebro, A. J. (2004). Advances in parallel heterogeneous genetic algorithms for continuous optimization. *Int. J. Appl. Math. Comput. Sci.*, 14:317–333.

Alba, E. and Tomassini, M. (2002). Parallelism and evolutionary algorithms. *IEEE Trans. on Evolutionary Computation*, 6(5):443–462.

Baker, J. E. (1987). Reducing bias and inefficiency in the selection algorithm. In Grefenstette, J. J., editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 14–21, Hillsdale, NJ. Lawrence Erlbaum Associates.

Brandao, J. (2004). A tabu search algorithm for the open vehicle routing problem. *European Journal of Operational Research*, 157(3):552–564.

Cantu-Paz, E. (2000). Markov chain of parallel genetic algorithms. *IEEE. Trans. Evolutionary Computation*, 4:216–226.

Chakraborty, U. K., Deb, K., and Chakraborty, M. (1996). Analysis of selection algorithms: A markov chain approach. *Evol. Comput.*, 4(2):133–167.

De Jong, K. A. (1975). *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, University of Michigan.

De Jong, K. A. (1996). Evolutionary computation: Recent developments and open issues. In Goodman, E. D., Punch, B., and Uskov, V., editors, *Proceedings of the First International Conference on Evolutionary Computation and Its Applications*, pages 7–17, Moscow.

Deb, K. and Goel, T. (2001). Controlled elitist non-dominated sorting genetic algorithms for better convergence. In *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 67–81. Springer-Verlag.

DellÁmico, M. and Trubian, M. (1993). Applying tabu search to the job-shop scheduling problem. *Ann. Oper. Res.*, 41(1-4):231–252.

Eirola, E., Liitiäinen, E., Lendasse, A., Corona, F., and Verleysen, M. (2008). Using the delta test for variable selection. In *ESANN 2008, European Symposium on Artificial Neural Networks, Bruges (Belgium)*, pages 25–30.

Eshelman, L. and Schaffer, J. (1993). Real-coded genetic algorithms and interval schemata. In Darrell Whitley, L., editor, *Foundation of Genetic Algorithms 2*, pages 187–202. Morgan-Kauffman Publishers, Inc.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.*, 13(5):533–549.

Glover, F. (1989). Tabu search part i. *ORSA Journal on Computing*, 1(3):190–206.

Glover, F. (1990). Tabu search part ii. *ORSA Journal on Computing*, 2:4–32.

Glover, F. (2006). Parametric tabu-search for mixed integer programs. *Comput. Oper. Res.*, 33(9):2449–2494.

Glover, F. and Laguna, F. (1997). *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, USA.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Addison Wesley.

Grefenstette, J. J. (1981). Parallel adaptive algorithms for function optimization. Technical Report TCGA CS-81-19, Department of Engineering Mechanics, University of Alabama, Vanderbilt University,.

Guillén, A., Pomares, H., González, J., Rojas, I., Herrera, L. J., and Prieto, A. (2008). Parallel multi-objective memetic rbfnns design and feature selection for function approximation problems. In *Neurocomputing, In press*.

Guillen, A., Rojas, I., Rubio, G., Pomares, H., Herrera, L., and Gonzalez, J. (2008). A new interface for mpi in matlab and its application over a genetic algorithm. In *Proceedings of the European Symposium on Time Series Prediction*, pages 37–46.

Guyon, I., Gunn, S., Nikravesh, M., and Zadeh, A. (2006). *Feature extraction: Foundations and applications (Studies in fuzziness and soft computing)*. Springer-Verlag New York, Secaucus, NJ, USA.

Hedar, A.-R. and Fukushima, M. (2006). Tabu search directed by direct search methods for nonlinear global optimization. *European Journal of Operational Research*, 170(2):329–349.

Herrera, L., Pomares, H., Rojas, I., Verleysen, M., and Guillen, A. (2006). Effective input variable selection for function approximation. *Lecture Notes in Computer Science*, 4131:41–50.

Holland, J. J. (1975). *Adaption in natural and artificial systems*. University of Michigan Press.

Ishibuchi, H., Yoshida, T., and Murata, T. (2003). Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Trans. on Evolutionary Computation*, 7:204–223.

Jones, A. (2004). New tools in non-linear modelling and prediction. *Computational Management Science*, 1(2):109–149.

Kosko, B. (1994). Fuzzy systems as universal approximators. *Computers, IEEE Transactions on*, 43(11):1329–1333.

Lee, S.-W. and Verri, A., editors (2002). *Pattern recognition with support vector machines, First International Workshop, SVM 2002, Niagara Falls, Canada, August 10, 2002, Proceedings*, volume 2388 of *Lecture Notes in Computer Science*. Springer.

Mantawy, A. H., Soliman, S. A., and El-Hawary, M. E. (2002). A new tabu search algorithm for the long-term hydro scheduling problem. *Power Engineering 2002 Large Engineering Systems Conference on, LESCOPE 02*, pages 29–34.

Mateo, F. and Lendasse, A. (2008). A variable selection approach based on the delta test for extreme learning machine models. In *Proceedings of the European Symposium on Time Series Prediction*, pages 57–66.

Michalewicz, Z. (1996). *Genetic algorithms + Data structures = Evolution programs*. Springer-Verlag, 3rd edition.

Mitchell, M. and Forrest, S. (1995). Genetic algorithms and artificial life. *Artificial Life*, 1(3):267–289.

Oh, I.-S., Lee, J.-S., and Moon, B.-R. (2002). Local search-embedded genetic algorithms for feature selection. *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, 2:148–151 vol.2.

Oh, I.-S., Lee, J.-S., and Moon, B.-R. (2004). Hybrid genetic algorithms for feature selection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(11):1424–1437.

Pi, H. and Peterson, C. (1994). Finding the embedding dimension and variable dependencies in time series. *Neural Computation*, 6(3):509–520.

Poggio, T. and Girosi, F. (1989). A theory of networks for approximation and learning. Technical Report AI-1140, MIT Artificial Intelligence Laboratory, Cambridge, MA.

Punch, W. F., Goodman, E. D., Pei, M., Chia-Shun, L., Hovland, P., and Enbody, R. (1993). Further research on feature selection and classification using genetic algorithms. In Forrest, S., editor, *Proc. of the Fifth Int. Conf. on Genetic Algorithms*, pages 557–564, San Mateo, CA. Morgan Kaufmann.

Rasmussen, C. E. (2004). *Gaussian processes in machine learning*, volume 3176 of *Lecture Notes in Computer Science*, pages 63–71. Springer-Verlag, Heidelberg. Copyright by Springer.

Raymer, M., Punch, W., Goodman, E., Kuhn, L., and Jain, A. (2000). Dimensionality reduction using genetic algorithms. *Evolutionary Computation, IEEE Transactions on*, 4(2):164–171.

Reeves, C. R. (1993). Using genetic algorithms with small populations. In Forrest, S., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 92–99. Morgan Kaufmann.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408.

Saeys, Y., Inza, I., and Larranaga, P. (2007). A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517.

Scheuerer, S. (2006). A tabu search heuristic for the truck and trailer routing problem. *Comput. Oper. Res.*, 33(4):894–909.

Sywerda, G. (1989). Uniform crossover in genetic algorithms. In *Proceedings of the third international conference on Genetic algorithms*, pages 2–9, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Wang, L. and Kazmierski, T. (2005). Vhdl-ams based genetic optimization of a fuzzy logic controller for automotive active suspension systems. *Behavioral Modeling and Simulation Workshop, 2005. BMAS 2005. Proceedings of the 2005 IEEE International*, pages 124–127.

Xu, J., Chiu, S., and Glover, F. (1996a). A probabilistic tabu search for the telecommunications network design. *Journal of Combinatorial Optimization, Special Issue on Topological Network Design*, 1:69–94.

Xu, J., Chiu, S., and Glover, F. (1996b). Using tabu search to solve steiner tree-star problem in telecommunications network design. *Telecommunication Systems*, 6:117–125.

Zhang, C., Li, P., Guan, Z., and Y. Rao, Y. (2007). A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem. *Computers & Operations Research*, 34(11):3229–3242.

Zhang, J., Li, S., and Shen, S. (2006). Extracting minimum unsatisfiable cores with a greedy genetic algorithm. In *Proc. ACAI06*, pages 847–856.

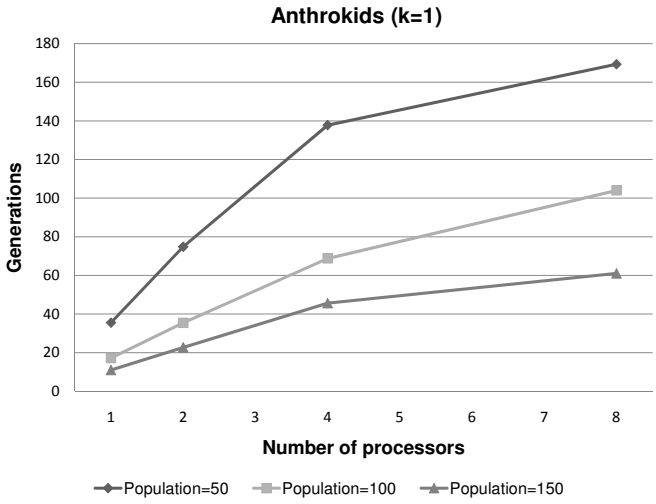

Figure 2: Generations evaluated by the GA vs the number of processors used. Anthrokids without scaling.

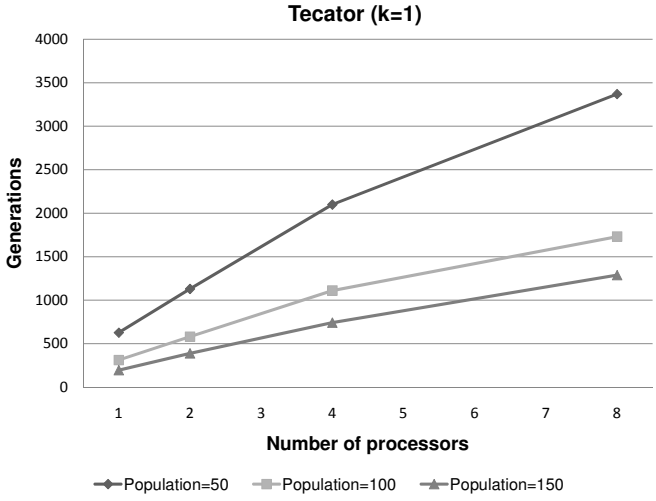

Figure 3: Generations evaluated by the GA vs the number of processors used. Tecator without scaling.

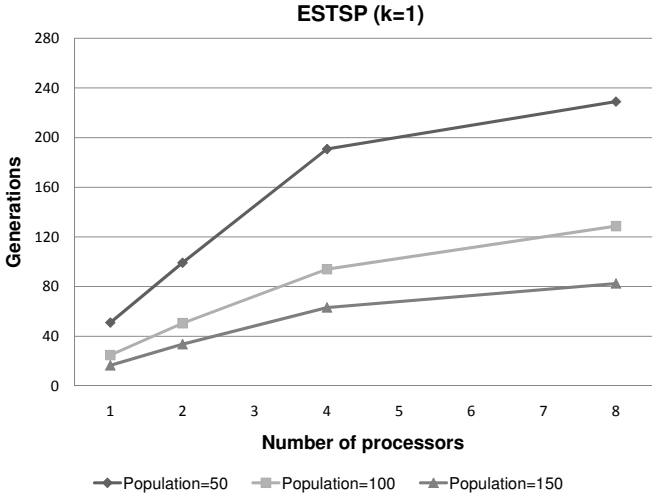

Figure 4: Generations evaluated by the GA vs the number of processors used. ESTSP without scaling.

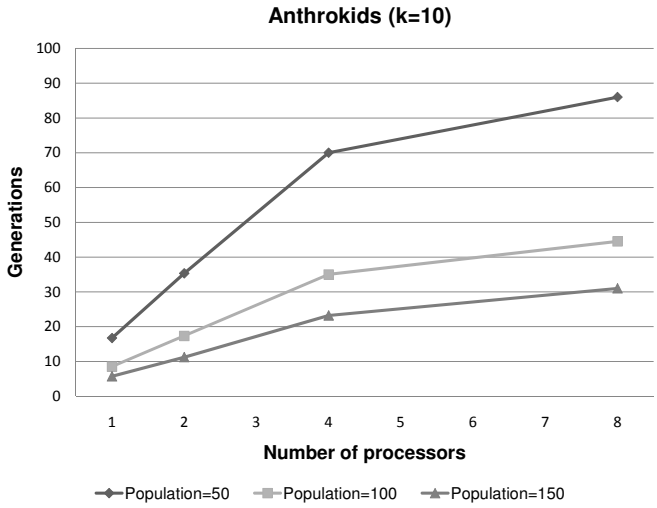

Figure 5: Generations evaluated by the GA vs the number of processors used. Anthrokids with scaling.

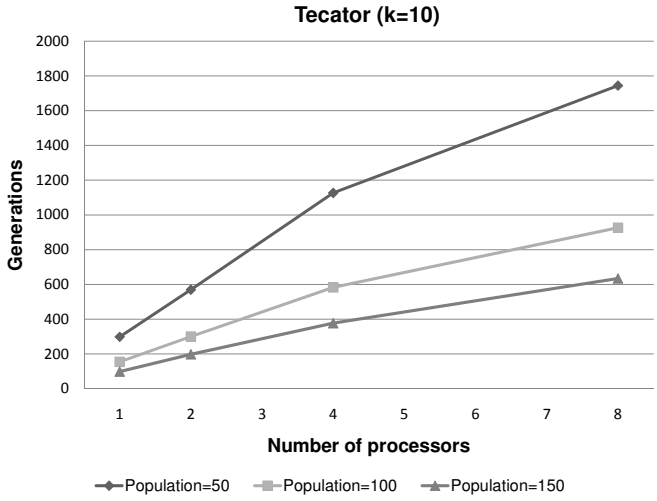

Figure 6: Generations evaluated by the GA vs the number of processors used. Tecator with scaling.

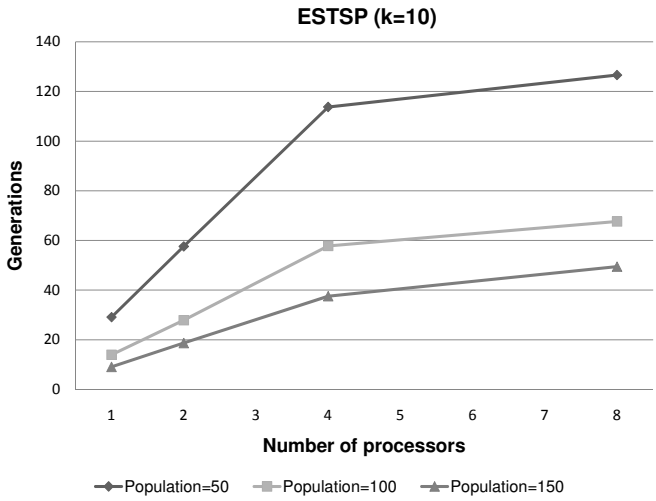

Figure 7: Generations evaluated by the GA vs the number of processors used. ESTSP with scaling.