Using Linear Regression to Predict Changes in Evolutionary Algorithms dealing with Dynamic Environments

Anabela Simões, Ernesto Costa
Using Linear Regression to Predict Changes in Evolutionary Algorithms dealing with Dynamic Environments

Anabela Simões¹,², Ernesto Costa²

¹Dept. of Informatics and Systems Engineering
ISEC - Coimbra Polytechnic
Rua Pedro Nunes - Quinta da Nora
3030-199 Coimbra – Portugal

²Centre for Informatics and Systems of the University of Coimbra
Pólo II – Pinhal de Marrocos
3030 - 290 Coimbra – Portugal
abs@isec.pt, ernesto@dei.uc.pt

CISUC TECHNICAL REPORT TR 2007/005 - ISSN 0874-338X

April 2007

Abstract. Many real-world problems change over time and usually, the moment when next change will happen is unknown. Evolutionary Algorithms have been widely used to deal with changing environments and the algorithm is constantly monitoring for alterations and just after detecting one some action is taken. Nevertheless, some of the studied environments are characterized by the periodicity of the change. In these cases it is possible to predict when the next change will occur and start using some mechanisms before the change take place. In this report we carried out an investigation in cyclic changing environments with periodic changes, using linear regression to predict when next change will occur. Based on the predicted value, the algorithm starts preparing the population for the near change. This idea is tested in a memory-based EA using a population and memory of variable sizes previously studied with considerable success. We assume that the predicted moment can have a small error. Before the change occurs two different actions can be taken in order to avoid the decrease of the algorithm’s performance under the new conditions. The results show that prediction is useful for cyclic environments when the period between changes is small and when more different states can appear in the environment.

Keywords: Evolutionary Algorithms, Dynamic Environments, Prediction, Memory, Statistics, Linear Regression

1 Introduction

Most real-world problems change over time and traditional Evolutionary Algorithms (EAs) are not suited to solve them. In these cases, some improvements have been introduced in EAs, such as the addition of memory or the use of strategies to promote the population’s diversity.

The main purpose of the memory-based approaches is to gather information from the past that can be useful in the future. The rationale behind promoting the population’s diversity is to avoid the convergence of the complete population towards a point of the search space, which may difficult its readaptation if a change happens. Nevertheless, most of evolutionary-based approaches used in changing optimization problems wait for the observation of an environmental alteration to decide what to do: use the information of memory [1], [7], [9], [12], [19], or change the rate of mutation in order to increase population’s diversity [4], for instance. Usually, the result is a reduction on the algorithm’s performance after a change, taking some time to readapt and find the new optimum.

When the environment is chaotic or completely random, we should not expect the EA to readapt to new situations immediately. But, assuming that the environment has some predictability, such as periodic changes or a certain repeated pattern, it is reasonable to think that using information from the past it is possible to estimate future situations.

Recently, several studies concerning anticipation in changing environments using EAs have been proposed. The main goal of these approaches is to estimate future situations and so decide the algorithm’s behavior in the present. Since that information about the future is not available, this information is attained through learning from past situations.
that, in general, conjugation allows VMEA to achieve improved results [13], [14]. Conjugation will be used in this work as the main genetic operator of the EA.

3. Predicting with Linear Regression

3.1. Statistical Model for Linear Regression

Simple linear regression studies the relationship between a response variable $y$ and a single explanatory variable $x$. This statistical method assumes that for each value of $x$ the observed values of $y$ are normally distributed about a mean that depends on $x$. These means are usually denoted by $\mu_y$. In general the means $\mu_y$ can change according to any sort of pattern as $x$ changes. In simple linear regression is assumed that they all lie on a line when plotted against $x$ [10]. The equation of that line is:

$$\mu_y = \beta_0 + \beta_1 \cdot x$$

(1)

with intercept $\beta_0$ and slope $\beta_1$. This is the linear regression line and describes how the mean response changes with $x$. The observed $y$ values will vary about these means and it’s assumed that this variation, measured by the standard deviation, is the same for all the values of $x$.

Linear regression allows inferences not only for samples for which the data is known, but also for those corresponding to $x$’s not present in the data. Three inferences are possible: (1) estimate the slope $\beta_1$ and the intercept $\beta_0$ of the regression line; (2) estimate the mean response $\mu_y$ for a given value of $x$; (3) predict a future response $y$ for a given value of $x$.

In general, the goal of linear regression is to find the line that best predicts $y$ from $x$. Linear regression does this by finding the line that minimizes the sum of the squares of the vertical distances of the points from the line.

The estimated values for $\beta_0$ and $\beta_1$ called $b_0$ and $b_1$ are obtained using previous observations through equations (2) and (3). The intercept $b_0$ is given by:

$$b_0 = \bar{y} - b_1 \cdot \bar{x}$$

(2)

The slope $b_1$ is given by:

$$b_1 = \frac{\text{cr} \cdot s_y}{s_x}$$

(3)

where $\bar{y}$ is the mean of the observed values of $y$, $\bar{x}$ is the mean of the observed values of $x$, cr the correlation between $x$ and $y$ given by equation (4), $s_x$ and $s_y$ the standard deviations of the observed $x$ and $y$, respectively, given by equations (5).

$$\text{cr} = \frac{1}{n-1} \sum \left( \frac{x_i - \bar{x}}{s_x} \right) \left( \frac{y_i - \bar{y}}{s_y} \right)$$

(4)

$$s_x = \sqrt{\frac{1}{n-1} \sum (x_i - \bar{x})^2} \quad s_y = \sqrt{\frac{1}{n-1} \sum (y_i - \bar{y})^2}$$

(5)

with $n$, the number of previous observations from $x$ and $y$.

Once estimated the slope and intercept of the regression line, we can predict the value of $y$ (called by $\hat{y}$) from a given $x$ using the regression line equation [10]:

$$\hat{y} = b_0 + b_1 \cdot x$$

(6)

In the next section we will show an example.

Note that linear regression does not test whether the data are linear. It assumes that the data are linear, and finds the slope and intercept that make a straight line best fit the known data. If the error in measuring $x$ is large, more inference methods are needed.

3.2. Predicting Next Change

We will assume that the changes in the environment are periodic or following a certain pattern that is repeatedly observed. In these situations it makes sense to use linear regression to predict the generation when next change will take place, based on previous observations.

In our implementation, the first two changes of the environment are stored without prediction. After that, it is possible to start predicting the next change based on the previous observations.

Suppose that three observations were already made ($n = 3$):

<table>
<thead>
<tr>
<th>Obs</th>
<th>$x$</th>
<th>$y$</th>
</tr>
</thead>
</table>
4.2. Knapsack Problem
The benchmark used was the 0/1 dynamic knapsack. The knapsack problem is a NP-complete combinatorial optimization problem often used as benchmark. It consists in selecting a number of items to a knapsack with limited capacity. Each item has a value ($v_i$) and a weight ($w_i$) and the objective is to choose the items that maximize the total value, without exceeding the capacity of the bag:

$$\max v(x) = \sum_{i=1}^{n} v_i x_i$$  \hspace{1cm} (7)

subject to the weight constraint:

$$\sum_{i=1}^{n} w_i x_i < C$$  \hspace{1cm} (8)

We used a knapsack problem with 100 items using strongly correlated sets of randomly generated data constructed in the following way ([8], [18]):

- $w_i =$ uniformly random integer $[1, 50]$  \hspace{1cm} (9)
- $v_i =$ $w_i$ + uniformly random integer $[1, 5]$  \hspace{1cm} (10)
- $C =$ $0.6 \times \sum_{i=1}^{100} w_i$  \hspace{1cm} (11)

The fitness of an individual is equal to the sum of the values of the selected items, if the weight limit is not reached. If too many items are selected, then the fitness is penalized in order to ensure that invalid individuals are distinguished from the valid ones. The fitness function is defined as follows:

$$f(x) = \begin{cases} 
\sum_{i=1}^{n} v_i x_i, & \text{if } \sum_{i=1}^{n} v_i x_i \leq C \\
10^{-3} \times 10^{\frac{\sum_{i=1}^{n} v_i x_i}{500}}, & \text{otherwise} 
\end{cases}$$  \hspace{1cm} (12)

4.3. Experimental Setup
4.3.1. Algorithms’ parameters
Previous work ([13], [14]) compared VMEA with other evolutionary algorithms: the random immigrants’ algorithm (RIGA) [5], the memory-based immigrants GA (MIGA) [18] and the memory-enhanced GA (MEGA) [18]. The results proved that VMEA, generally outperformed the other approaches. The best results were achieved when VMEA used the generational replacing strategy and the conjugation operator [14].

We will compare VMEA enhanced with the prediction unit, using bacterial conjugation and the generational replacing method as well. This VMEA was run using the two possible actions formerly described: upd1 (VMEA upd1) and upd2 (VMEA upd2).

The algorithms’ parameters were set as follows: generational replacement with elitism of size one, tournament selection with tournament of size two, conjugation with probability $p_c = 0.7$ and mutation applied with probability $p_m = 0.1$. VMEA using prediction will start with a population of 100 individuals and a memory of 10 individuals. The sum of the two populations cannot go beyond 150 individuals, except in the intervals described in section 3.3. In this interval the allowed maximum raised to 200 individuals. VMEA was run as described in [13], using initial sizes for population and memory of 125 and 25 individuals, respectively. The sum of the two populations could not surpass the 200 individuals. The assumed error for the predicted values in $VMEA_p$, to identify the interval $[nextgen - error, nextgen + error]$, was 5 generations.

For each experiment of an algorithm, 30 runs were executed and the number of environmental changes was: (1) in environments changing every $r$ generations: 200 when $r = 10$ (2000 generations), 80 when $r = 50$ (4000 generations) and 40 when $r = 100$ and 200 (4000 and 8000 generations, respectively); (2) using the patterned periodic changes, the number of generations was generated based in 200 environmental changes. The overall performance used to compare the algorithms was the best-of-generation fitness averaged over 30 independent runs, executed with the same random seeds:

$$F_{overall} = \frac{1}{G} \sum_{n=1}^{G} \left( \frac{1}{N} \sum_{j=1}^{N} F_{genj} \right)$$  \hspace{1cm} (13)

$G =$ number of generations, $N =$ number of runs.
5.2. Performance of the Algorithms
In this section we will show the results obtained by the VMEA without the prediction module (VMEA) and by VMEA using prediction (VMEA\textsubscript{upd1} and VMEA\textsubscript{upd2}). The results are exposed in two separated sections: the first for changes in every \( r \) generation and another for changes determined by a pattern such as described in section 4.1. The statistical results comparing the algorithms are reported in tables 1 and 2. We used paired one-tailed t-test at a 0.01 level of significance. The notation used in tables 1 and 2, to compare each pair of algorithms is ‘+’, ‘-‘, ‘+\textasciitilde’ or ‘-\textasciitilde’, when the first algorithm is better than, worse than, significantly better than, or significantly worse than the second algorithm.

5.1.1. Results in cyclic environments with a fixed change period
The global results given by equation (13) are plotted in Figure 5. We can see that for small change periods, the prediction module introduced in the algorithm increased its performance. For larger change periods, all the algorithms had similar behavior, except when the change ratio was higher.

<table>
<thead>
<tr>
<th>Table 1. T-test results comparing VMEAs in cyclic environments with fixed change period = 10, 50, 100, 200</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="chart.png" alt="" /></td>
</tr>
<tr>
<td><strong>T-test results</strong></td>
</tr>
<tr>
<td>VMEA - VMEA\textsubscript{upd1}</td>
</tr>
<tr>
<td>VMEA - VMEA\textsubscript{upd2}</td>
</tr>
<tr>
<td>VMEA\textsubscript{upd1} - VMEA\textsubscript{upd2}</td>
</tr>
<tr>
<td>VMEA - VMEA\textsubscript{upd1}</td>
</tr>
<tr>
<td>VMEA - VMEA\textsubscript{upd2}</td>
</tr>
<tr>
<td>VMEA\textsubscript{upd1} - VMEA\textsubscript{upd2}</td>
</tr>
<tr>
<td>VMEA - VMEA\textsubscript{upd1}</td>
</tr>
<tr>
<td>VMEA - VMEA\textsubscript{upd2}</td>
</tr>
<tr>
<td>VMEA\textsubscript{upd1} - VMEA\textsubscript{upd2}</td>
</tr>
<tr>
<td>VMEA - VMEA\textsubscript{upd1}</td>
</tr>
<tr>
<td>VMEA - VMEA\textsubscript{upd2}</td>
</tr>
<tr>
<td>VMEA\textsubscript{upd1} - VMEA\textsubscript{upd2}</td>
</tr>
</tbody>
</table>

In this case, the introduction of the prediction module allowed faster adaptation of the algorithm to the observed alterations. Table 1 confirms these observations. We see that VMEA globally achieved the worst performances. In table 1 we can also see that the method \textit{upd2} used to prepare the population before the change was in general better than the \textit{upd1} method.

Figures 6 and 7 show some examples of how the VMEAs’ performance evolved through the time. It is clear that for smaller change periods the prediction unit was very helpful and increased the algorithm’s performance. For larger change periods (200) the improvements are visible for inferior change ratios, because in these cases the number of different states is superior and the memory in certain periods wasn’t enough to allow continuous adaptability.

5.1.2. Results in periodic environments following a pattern change
In the environments with periodic changes based on a repeated pattern the conclusions were similar to the previously described. In the patterns with smaller periods, 10-20-10 and 50-60-70, the VMEA using prediction obtained the best scores. In the pattern with larger periods, 100-150-100, the improvements introduced by the prediction method were not so visible. Table 2 and Figure 8 show the global results comparing all the algorithms in the three tested patterns. In all the analyzed cases the \textit{upd2} and \textit{upd1} methods obtained analogous results. Figures 9 and 10 show some examples of the dynamics of the algorithms along the time. All the conclusions stated before are once again visible in these plots.
5.1.3. Is prediction really useful?

To see if the prediction unit was responsible for the observed improvements, we run VMEA without the prediction module, but using the upd1 and upd2 methods after the change occur. So, instead of using the scheme of the original VMEA (retrieve from the memory the best individual and introduce it into the population), we allowed the global number of individuals to increase up to 200 in the moment after the change, merging the memory with the population (upd1) or creating some new individuals applying conjugation in the memory and then merging this individuals with the population and with part of the memory (upd2).

We observed that using these schemes the VMEA’s performance was increased, when compared with the original VMEA, but the algorithms using the prediction unit were in fact the most proficient. Figures 11 and 12 show some of the results comparing VMEA with no prediction and VMEA without prediction and using the upd1 and upd2 schemes (applied after the change). So, it seems that the prediction unit was in fact responsible for the observed improvements. Preparing the population before the change happens proved to be helpful to the readaptation of the algorithm. Currently we are trying to use different methods before the change occur, such us, estimating also the direction of the change and introducing individuals that probably will be the fittest after the next change happens.


