Prediction in Evolutionary Algorithms for Dynamic Environments Using Markov Chains and Nonlinear Regression

Anabela Simões *
Coimbra Polythechnic
Rua Pedro Nunes - Quinta da Nora
3030-199 Coimbra, Portugal
abs@isec.pt

Ernesto Costa
CISUC, University of Coimbra
Polo II - Universidade de Coimbra
3030-290 Coimbra - Portugal
ernesto@dei.uc.pt

ABSTRACT

The inclusion of prediction mechanisms in Evolutionary Algorithms (EAs) used to solve dynamic environments allows forecasting the future and this way we can prepare the algorithm to the changes. Prediction is a difficult task, but if some recurrence is present in the environment, it is possible to apply statistical methods which use information from the past to estimate the future. In this work we enhance a previously proposed computational architecture, incorporating a new predictor based on nonlinear regression. The system uses a memory-based EA to evolve the best solution and a predictor module based on Markov chains to estimate which possible environments will appear in the next change. Another prediction module is responsible to estimate when next change will happen. In this work important enhancements are introduced in this module, replacing the linear predictor by a nonlinear one. The performance of the EA is compared using no prediction, using predictions supplied by linear regression and by nonlinear regression. The results show that this new module is very robust allowing to accurately predicting when next change will occur in different types of change periods.

Categories and Subject Descriptors

I. [**Computing Methodologies**]: ARTIFICIAL INTELLIGENCE—*Problem Solving, Control Methods, and Search*

General Terms

Algorithms, Experimentation, Performance

Keywords

Evolutionary Algorithms, Dynamic Environments, Prediction

*also belongs to CISUC

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GECCO'09, July 8–12, 2009, Montréal Québec, Canada.
Copyright 2009 ACM 978-1-60558-325-9/09/07 ...$5.00.

1. INTRODUCTION

Evolutionary Algorithms (EA) have been successfully used to solve different types of optimization problems in static environments. When the environment changes along time, the optimum we are looking for will, in general, be different. Due to its characteristics the algorithm will have difficulties to redirect the search in order to find the new optimum, or near optimum, in acceptable time. That is why some modifications were proposed to cope with that problem: the use of memory to keep track of past good solution's candidates, techniques to maintain a high level of population's diversity or mechanisms relying on different populations, each one tuned for a particular environment (see [17] for a review on those alternatives). In certain cases, the dynamics of the environment's changes are not chaotic but follows instead a certain repeating pattern and, in those cases, we may expect to be able to predict both the **time** and the **trend** of the modification. If we succeed on that endeavor we may avoid the sudden decrease in performance that typically results from the change in the environment improving thus the algorithm's adaptability. In this paper we extend some ideas we develop recently and which are based on two predictors: one, based on regression, to estimate **when** the next change will take place and, the other one, supported by a Markov chain model, to predict **how** the environment will be different. The main goal of this paper will be to evaluate experimentally a predictor based on **nonlinear regression** and to compare its performance with one supported by a linear regression mechanism. They will be combined with the predictor involving a **Markov Chain** and will be tested in connection with different types of dynamic environments. The remaining text is organized as follows: section 2 describes related work concerning prediction and anticipation used by EAs in the context of dynamic environments. In section 3 we explain the overall architecture of an EA that utilizes the two prediction modules. The nonlinear regression predictor is explained in section 4. In section 5 we present the experimental setup used to test the investigated ideas. Experimental results are summarized in section 6. We conclude with some remarks and ideas for future work.

2. RELATED WORK

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 information about the future typically is not available, it is attained through learning from past situations. Branke et al. [4] try to understand how the decisions made at one stage influence the problems encountered in the future. Future changes are anticipated by searching not only for good solutions but also for solutions that additionally influence the state of the problem in a positive way. These so-called flexible solutions are easily adjustable to changes in the environment. Studies on the tardiness job-shop problem, with jobs arriving no-deterministically over time, showed that avoiding early idle times increases flexibility, and thus the inclusion of an early idle time penalty as secondary objective into the scheduling algorithm can significantly enhance the system's performance. Stroud [15] used a Kalman-Extended Genetic Algorithm (KGA) in which a Kalman filter is applied to the fitness values associated with the individuals that make up the population. This is used to determine when to generate a new individual, when to re-evaluate an existing individual, and which one to re-evaluate. This KGA is applied to the problem of maintaining a network configuration with minimized message loss in which the nodes are mobile and the transmission over a link is stochastic. As the nodes move, the optimal network changes, but information contained within the population of solutions allows efficient discovery of better-adapted solutions. Van Hemert et al. [16] introduced an EA with a meta-learner to estimate at time t how the environment will be at time $t+\delta$. This approach uses two populations, one that searches the current optimum and another that uses the best individuals in the past to predict the future best value. The prediction is made based on observations from the past using two types of predictors: a perfect predictor and a noisy predictor. In reality they should not be called predictors. Concerning the former, the correct optimal value at the future time step **is given to the solver**, and for the latter the noisy predictor just provides the system noisy values as the optimal solution for the next step. The idea was tested with two benchmark problems: the knapsack problem and the Ŏsmera's function. Bosman [1],[2] proposed in the last years several approaches focused on the importance of using learning and anticipation in online dynamic optimization. These works analyze the influence of time-linkage present in problems such as scheduling and vehicle routing. The presence of time-linkage in this kind of problems can influence the overall performance of the system: if a decision is made just to optimize the score at a specific moment, it can negatively influence the results obtained in the future. Bosman's works propose an algorithmic framework integrating evolutionary computation with machine learning and statistical learning techniques to estimate future situations. Predictions are made based on information collected from the past. The used predictor is a learning algorithm that approximates either the optimization function or several of its parameters. Rossi et al. [7] compare different techniques to improve the search for tracking a moving optimum using the information provided by a predictor mechanism using Kalman filters. The used predictor assumes that the changes in the environment are not random and can be learned, helping the EA to keep track of the current optimum. The use of linear regression to predict the moment of next change was initially proposed by Simões and Costa [11]. The idea was tested with different dynamic optimization problems, using a variable-size memory EA [12]. Several issues were analyzed such as the speed or the severity of change. The results showed that, if some pattern can be found in the changes of the environment, the predictor gives accurate estimations that can be used to enhance the EA's adaptability to future situations [11]. Later, in [13] a predictor based on Markov chain was added and used to predict which environments may appear in the future.

3. USING PREDICTION IN THE EVOLUTIONARY ALGORITHM

Simões and Costa [13] proposed a computational model called **PredEA** to deal with dynamic environments. The proposed architecture uses a traditional EA that evolves a population of individuals that aim to optimize the current fitness function. A memory is used to store useful information from the past that is used in future changes. The traditional memory-based EA was extended with two prediction modules. The first module uses information about when previous changes have occurred to estimate the generation when the next change will be observed. In previous work, predictions provided by this module were made by a linear regression predictor [13]. The second, using Markov chains, keeps track of previous environments and provides predictions about which environments will appear in the future. A different module (anticipation) uses the information provided by the previous two modules and prepares the EA for the next change. Figure 1 illustrates the proposed architecture.

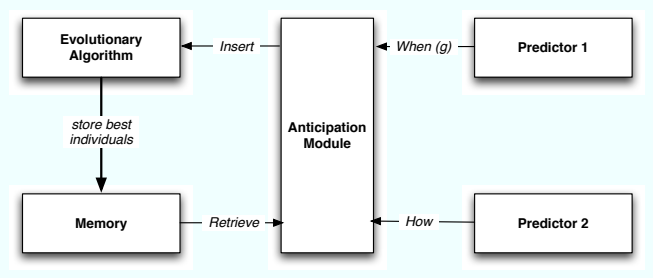

Figure 1: Computational architecture

Here is a brief description of the components:
*** Evolutionary Algorithm:** standard evolutionary algorithm which evolves a population of individuals through the application of selection, crossover and mutation.
*** Memory:** the memorized individuals are associated with the state (environment) where they were the best solution. It is updated from time to time with the current best individual of population.
*** Predictor 1 (P1):** this component uses previous information to predict **when** next change may occur. In previous work this module used a linear regression predictor. In this paper we will propose and test a nonlinear based predictor to this module.
*** Predictor 2 (P2):** every time a different environment appears this module stores the environmental information. It consists in a set of states (each state corresponds to a different environment), a matrix of state transition probabilities and the initial probability vector. Each state corre-

sponds to a different environment. The initial probability vector is initialized choosing randomly the initial state. The state transition probability matrix starts filled with zeros and is updated *on-the-fly* when different environments appear. When this module is called, it uses all the available information to estimate **which environment(s)** will appear next.

*** Anticipation module (A)**: manages all the information provided by the two predictors and decides when to activate the mechanisms to prepare the EA to the next change. At that time, information from memory is retrieved and inserted into the population. This information corresponds to those individuals that can be useful to the next predicted environment(s). If the **P2** module doesn't provide any prediction, five random individuals from memory are inserted into the population, replacing five randomly selected.

The detailed description of each one of these modules can be found in [13].

When used with a linear regression predictor the module **P1** works correctly if the change period follows a linear (or close to linear) trend [13]. For instance, if the environment changes in a periodic manner, every r generations, then the predicted values are precise. If a different pattern is observed in the change period, there will be an error associated to the values provided by the $P1$ module. A control parameter called Δ was used in the algorithm to cover that possible error and to decide **how many** generations **before** the predicted change the **A** module must be activated. Initially the value of Δ was chosen off-line and kept constant during the entire run. This method was a limitation to the predictor's efficacy and different methods of adjusting the value of Δ during the run were proposed and successfully tested [9]. The pseudocode of the **PredEA** is detailed in Figure 2.

The efficacy of the linear predictor can be compromised if the environment changes following a nonlinear trend. In these cases the value of the prediction error may be so large that the value of Δ leads to a significant Here, the $P1$ module is tested using a **nonlinear regression** predictor which will be able to make accurate predictions in environments changing periodically or following linear and nonlinear patterns.

PredEA_improved(max, markov, initial-state)

```
1.   Randomly create initial population
2.   Create empty memory
3.   Initialize Δ = 5
4.   Create the transition matrix with max sates filled
      with zeros
5.   repeat
6.     Evaluate population
7.     Evaluate memory
8.     if is time to update memory then
9.       Store best individual
10.      Set next time to update memory
11.    if an environmental change happens then
12.      Store performance measures
13.      Activate the P1 module
             i. Update P1 information
             ii. Predict g (next_change)
14.      Update the value of Δ
15.      Update the algorithm's Markov transition matrix
16.    if g (next_change) is close (g -Δ) then
17.      Activate the P2 module
             i. Predict next state(s)
18.      Activate the A module:
             i. Search memory for best individual(s)
             ii. Introduce those individuals into pop.
         %Standard EA steps
19.      Perform Selection, crossover and mutation
20.      Define next population
21.  until stop_condition
```

max is the maximum number of states of the Markov chain
markov is the Markov model defined off-line used in $P2$
initial-state is the randomly chosen initial state for the Markov model.

Figure 2: PredEA Pseudocode

4. NONLINEAR REGRESSION

Usually, linear regression is used to model relationships between variables that follow a linear correlation. Some nonlinear functions can be modeled using linear regression (e.g. polynomial regression) but nonlinear regression is often used in these cases because allows modeling a wide range of functions. In next sections we present the techniques for modeling data that displays nonlinear behaviors and use functions that are nonlinear in the model parameters.

4.1 Nonlinear Regression Basics

The basic idea of nonlinear regression is the same as that of linear regression, namely to relate a response y to a vector of predictor variables x. Nonlinear regression is characterized by the fact that the prediction equation depends nonlinearly on one or more unknown parameters. The basic form for a nonlinear model between the response y and a predictor x is given as:

$$y_i = f(x_i, \theta) + e_i \quad (1)$$

where y_i and x_i are the data, f is a nonlinear function involving the predictor and the parameter vector θ and e_i a random error [8]. For instance, let's assume the asymptotic regression model:

$$f(x) = \theta_1 - \theta_2\theta_3^x, \text{ where } 0 < \theta_3 < 1 \quad (2)$$

Figure 3 shows two examples of functions using different values for the parameters θ_i.

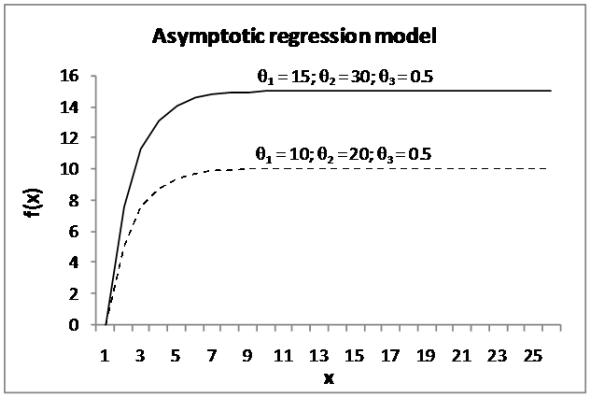

Figure 3: The asymptotic regression model

Although nonlinear regression is less intuitive and more complicated to use than linear regression, it is more powerful because allows predictions in both linear and nonlinear data

and are more suitable to model information of real world which is in general nonlinear. The difficult task in nonlinear regression is to estimate the correct values for the parameter vector θ. Once estimated the parameters, predictions can be performed using the nonlinear function. Next section briefly explores the techniques for estimating the nonlinear parameters.

4.2 Parameter Estimation

The task of parameter estimation for nonlinear regression is not straightforward. Usually, statistical software using numerical algorithms is used to analyze the data and produce the best parameter's choice for that data [8]. A nonlinear parameter estimation problem is an optimization problem which goal is to minimize the sum of squared errors given by eq. 3:

$$Sum_{e_i} = \sum_{i=1}^{n} (y_i - f(x_i, \theta_i))^2 \quad (3)$$

Rather than minimizing the sum of squared errors, other techniques minimize the sum of absolute deviations. Several function minimization methods are used in parameter estimation, for instance, weighted least squares, maximum likelihood, Quasi-Newton method, Simplex procedure or Hooke-Jeeves pattern moves [8]. In general, these methods are not easily controllable and require much auxiliary information to work correctly. Another option, more general and easy to apply, is to use a genetic algorithm to evolve a population of individuals that minimize an objective function.

4.3 Nonlinear Regression in PredEA

In this work we propose a new method, which works with nonlinear regression, to use in the $P1$ module showed in Figure 1. Using a nonlinear based predictor we achieved more robust and accurate predictions for the moment of next change in a wider range of situations. The $P1$ module has a set of functions f_1, f_2, ..., f_n, that can be used to give the predictions. At time t, only one function is active. The choice of the active function is made measuring the prediction errors of all functions. The function with lower error is the selected one. The vector parameter θ_i is estimated using a standard GA as proposed by [6]. This GA uses the available information from the past to find the values for θ_i which minimize the errors of eq. 3. Every time a change occurs in the environment and additional information is available, the GA is executed to find a vector θ that better fits the data. The GA evolves a population of binary strings which correspond to different values for θ. The fitness function that the GA has to minimize is the least squares error function (eq. 3). Because individuals with higher fitness are selected more often, after some generations the best individual represents the optimal solution for θ. The vector θ is estimated using only the known data. Using these estimated parameters and the selected function, the $P1$ module predicts when next change will occur. After the real change happens, the prediction error is computed. If the error is superior to an established threshold α, the $P1$ module analyzes all the available functions to see if this error can be reduced. If so, a different function is used for future predictions. These actions correspond to the step 13i. of the algorithm given in Fig. 2. Figure 4 shows how the proposed module works. In the beginning, the first function is selected randomly.

5. EXPERIMENTAL DESIGN

The algorithm described before was tested and compared with similar algorithms: an EA without any prediction mechanism, and the **PredEA** using linear regression in the $P1$ module. The identification of these three algorithms will be:
*** PredEA-NLR**: memory-based evolutionary algorithm using predictors based on Markov chains and nonlinear regression;
*** PredEA-LR**: memory-based evolutionary algorithm using predictors based on Markov chains and linear regression;
*** NoPredEA**: memory-based evolutionary algorithm without prediction mechanisms.
Due to space restrictions we could not include all the details and results in this paper. More information and results about next sections can be found in [14].

5.1 Benchmark Problems

The benchmarks used to test the proposed method were the dynamic bitmatching problem (DBM) and the dynamic knapsack problem (DKP). The first problem can be described as: given a binary template, the individual's fitness is the number of bits matching the specified template. The knapsack problem consists in selecting a number of items to a knapsack with limited capacity. Each item has a value and a weight and the objective is to choose the items that maximize the total value, without exceeding the capacity of the bag. When it is time to change the environment, the template or the capacity of the bag are changed.

5.2 Experimental Setup

5.2.1 Parameters of the Evolutionary Algorithm

The EA's parameters were set as follows: generational replacement with elitism of size one, tournament selection with tournament of size two, uniform crossover with probability $p_c = 0.7$ and flip mutation applied with probability $p_m = 0.01$. Binary representation was used with chromosomes of size 100 (size of the binary templates and number of items for the knapsack problem). Population of 80 individuals and a memory of 20 individuals were used. The memory update times are made as suggested in [17] and the individuals are stored in the memory using the *generational* replacing strategy proposed in [10]. This scheme optimizes de capacity of the memory storing the best individual found

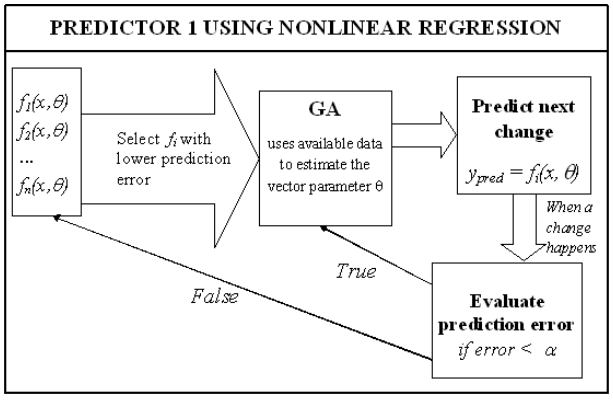

Figure 4: $P1$ module using nonlinear regression

so far for the present environment. When the memory becomes full, the similar individual is replaced by the best individual in the main population.

5.2.2 Parameters of the GA for nonlinear regression parameter's estimation

The GA used to estimate the nonlinear regression parameters was run for 100 generations, using standard parameter settings: population of 30 individuals, generational replacement with elitism of size one, tournament selection with tournament of size two, one-point crossover with probability p_c=0.75 and flip mutation applied with probability p_m=0.05 (as suggested in [6]).

5.2.3 Parameters of the $P1$ module

The nonlinear predictor was used with two functions which can model both linear and nonlinear patterns.

$$f_1(x) = \theta_1 x^2 + \theta_2 x + \theta_3 \quad (4)$$

$$f_2(x) = \frac{\theta_1 x}{\theta_2 + \theta_3 x} \quad (5)$$

The results presented in section 6 were obtained using the Δ adjusting method called Max_Av_Err [9] either in **PredEA-NLR** and **PredEA-LR**. This method updates the value of Δ using the maximum prediction error and the average of the positive prediction errors and was the approach that allowed the EA to obtain the best results.
The parameter α, used to choose which function will make the predictions was set to 10. If the predicted error is superior to α, a new evaluation of the model is done and the module can change the active function if its application reduces the prediction error.

5.2.4 Parameters of the $P2$ module (Markov chain)

The number of different states (templates or capacities) used in the experimentation were 3, 5, 10, 20 and 50. The environmental transitions were of two kinds: deterministic, i.e. the probability to change to the next state is always 1 or probabilistic, where, in certain states, the transition can be made to different states. The different states corresponding to the different templates for the dynamic bitmatching problem or to the different capacities of the knapsack problem were generated off-line and randomly selected every time a change in the environment happens. More details can be found in [13].

5.2.5 Types of Change Period

Three different types of change period were used: periodic, patterned and nonlinear.
Periodic (linear) This type of change period follows a linear trend. If the change period is set to r, the environment changes every r generations. We used $r = 10$, $r = 50$, $r = 100$ and $r = 200$.
Patterned (close to linear) In this case, the change period is generated through the repetition of a determined pattern. A pattern is set and the moments of change are calculated based on that pattern. Four different patterns were investigated: 5-10-5, 10-20-10 (fast), 50-60-70 (medium) and 100-150-100 (slow). This way the intervals between changes are not always the same, but the global behavior is *close to linear*. For instance if the pattern 5-10-5 is used, the first change occurs at generation $g_1 = 5$, the second at generation $g_2 = 5+10$, the third at generation $g_3 = 15+5$, the i^{th} generation at $g_i = g_{i-1} + k_j$, with $j = 1$ or $j = 2$. $k_1 = 5$ if i corresponds to a odd change number and $k_2 = 10$ otherwise.
Nonlinear The two nonlinear change periods were generated using f_1 and f_2 (eq. 4 and eq. 5) with the following parameters: $\theta_1 = 60$, $\theta_2 = 2$ and $\theta_3 = 0.1$ for f_1 and $\theta_1 = 62.67$, $\theta_2 = 0.627$ and $\theta_3 = 0.0047$ for f_2. This information is **unknown** to the prediction module $P1$, which evolves the values of θ_i through a GA using only the known data. These two functions allow to model different behaviors for the change period. In the first (f_1), the environment changes slower at the beginning and it becomes faster along time. Using f_2 the changes in the environment occur faster. These types of change period will be referred as **NL1** and **NL2**, respectively. Figure 5 shows the generations where the environment changes using f_1 and f_2.

5.2.6 Performance Measures

For each experiment, 30 runs were executed and the number of generations was computed based on 500 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. To observe the algorithm's performance along time we used the off-line performance. Off-line performance is calculated as the average of the best values found so far at each time step [3]. Only the individuals evaluated since the last change are considered.

6. RESULTS

In this section we will show some of the obtained results concerning the efficacy of the predictors and the performance

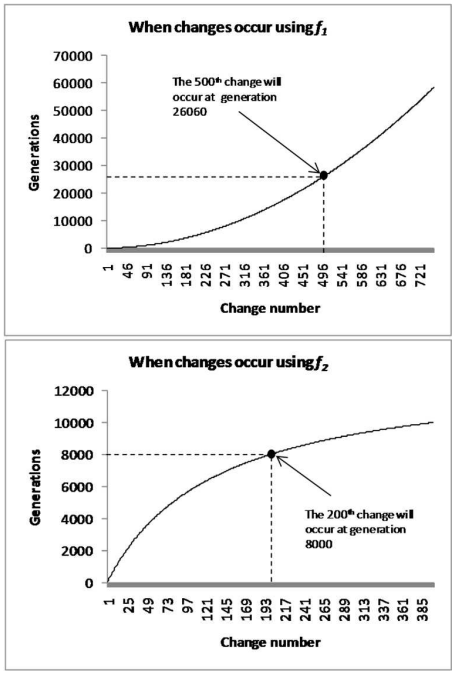

Figure 5: The change period follow a nonlinear pattern

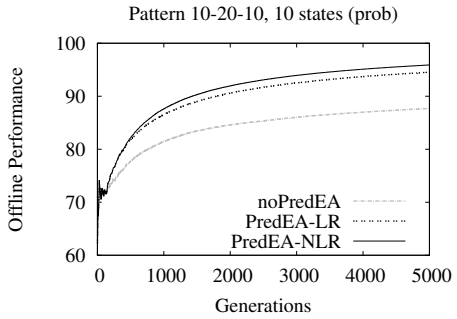

Figure 6: Off-line performance of PredEA, pattern 10-20-10, 10 states, probabilistic, DBM

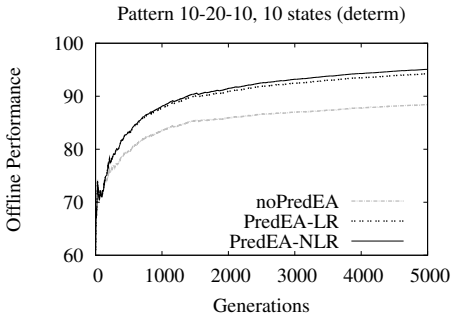

Figure 7: Off-line performance of PredEA, pattern 10-20-10, 10 states, deterministic, DBM

of the different algorithms. The results were statistically validated and the major statistical information of comparing the different methods can be found in [14]. We used paired one-tailed t-test at a 0.01 level of significance [5]. The statistical data support that the proposed predictor is efficient and robust and improved the algorithm's performance.

6.1 Prediction Efficacy

The efficacy of prediction was measured using the total number of changes observed and the response of the algorithm to those changes. If the algorithm was able to provide useful information ***before*** the next change effectively happens and ***after*** the previous change, then we consider that the prediction was accurate, otherwise it is considered inaccurate. Table 1 shows the accuracy of the module $P1$ using linear regression and the proposed nonlinear regression predictor. The average of the prediction errors and the size of Δ are also shown. As we can see in Table 1 the proposed predictor based on nonlinear regression was able to provide accurate predictions in all types of change periods. For the situations where the changes occurred in a nonlinear trend and the linear predictor failed, the nonlinear predictor successfully estimated the next change. The values reported in Table 1 also show that the prediction error was smaller using the nonlinear based predictor. This means that the estimated values where equal or better than the values provided by the linear regression predictor. The value of Δ was also improved. The smaller values for Δ obtained by the nonlinear regression predictor mean that the algorithm was able to react closer to the change in the environment, saving computational effort.

6.2 PredEA's Performance

Table 2 shows the global performance of the algorithm solving the DBM problem using different types of change periods and different number of environments (states). The best scores are marked with bold. The results show that the algorithm using the nonlinear predictor (**PredEA-NLR**) obtained the best results. In the change periods following a nonlinear trend the results were significantly improved when compared with the **PredEA-LR** algorithm. This is more visible in the **NL2** type. In this case the evironmental changes occur faster and the EA without prediction may have some difficulties in readapting after the change. Also, this was the case where the linear regression predictions lead to an untimely use of the information given by the $P2$ module. In this case, the information was always introduced into the population **after** the change happens. The new method avoids this error and the **PredEA**'s performance was improved. The results obtained in the dynamic DKP were analogous (see [14]). Figures 6 to 9 show the evolution of the algorithms' performance along time. The plots show the off-line performance obtained by the EA with and without prediction solving the the dynamic bitmatching problem. We can see that the **PredEA** using the new method

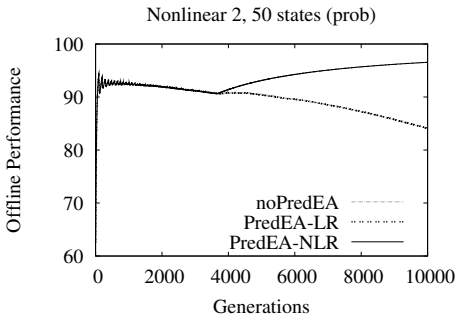

Figure 8: Off-line performance of PredEA, NL2 change period, 50 states, probabilistic, DBM

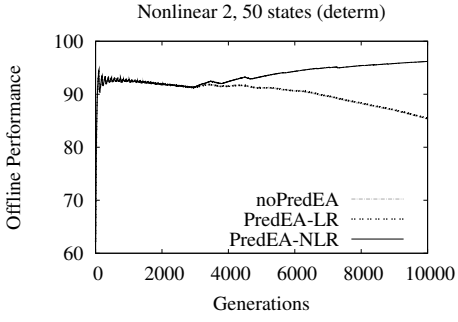

Figure 9: Off-line performance of PredEA, NL2 change period, 50 states, deterministic, DBM

based on nonlinear regression (**PredEA-NLR**) was able to improve the algorithm's performance. By the reasons stated before, this enhancement was more considerable in the **NL2** change period. In Figures 8 and 9 the **PredEA-LR** and **no-PredEA** curves overlap.
More results can be consulted in [14]. All the results support that the proposed predictor is efficient and robust and improved the algorithm's performance. It is capable of accurately predict when next change will occur in environments changing periodically, following a repeated pattern or a nonlinear curve.

7. CONCLUSIONS

In this paper we propose a new, precise and robust prediction method to use in EA dealing with dynamic environments. This predictor, which uses nonlinear regression, is responsible to estimate when the next change in the environment will happen. In previous work this task was made by a predictor based on linear regression which was limitative in situations where the change period was not periodic or changes follow a nonlinear pattern. The proposed method overcomes these limitations and proved to be robust and efficient, allowing to make accurate predictions in a wider range of situations.
Analyzing the obtained results, some conclusions can be stated. First, the nonlinear regression predictor was able to accurately predict when next change will take place in all types of change periods studied. Second, the precision of the predictions was improved, especially in the situations where the linear regression predictor failed. Third, better predictions lead to a better adjustment of the value of Δ avoiding unnecessary and expensive computational efforts.
The major limitation of the proposed algorithm is the number of functions used in the predictor module $P1$. The efficacy of the EA can be compromised if the environment changes following a pattern that cannot be modeled by any of the existing functions. Some improvements are being introduced and tested to overcome this limitation, namely the incorporation of more functions and the use of a GP to evolve the function that best fits the known data.

8. ACKNOWLEDGMENTS

The work of the first author described in this report was partially financed by the PhD Grant BD/39293/2006 of the Foundation for Science and Technology of the Portuguese Ministry of Science and Technology and High Education.

9. REFERENCES

[1] P. A. N. Bosman and H. La Poutré. Computationally Intelligent Online Dynamic Vehicle Routing by Explicit Loa Prediction in Evolutionary Algorithm. In T. P. Runarsson, H.-G. Beyer, E. Burke, J. Merelo-Guervós, L. D. Whitley and X. Yao, editors, *Proceedings of Parallel Problem Solving from Nature (PPSN IX)*, Lecture Notes in Computer Science, volume 4193, pages 312–321. Springer-Verlag, 2006.

[2] P. A. N. Bosman and H. La Poutré. Inventory Management and the Impact of Anticipation in Evolutionary Stochastic Online Dynamic Optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2007)*, pages 268–275. IEEE Press, 2007.

[3] J. Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer Academic Publishers, 2002.

[4] J. Branke and D. Mattfeld. Anticipation in dynamic optimization: The scheduling case. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J. Merelo and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature*, pages 253–262. Springer, 2000.

[5] D. S. Moore and G. P. McCabe. *Introduction to the Practice of Statistics (4th edition)*. Freeman and Company, 2003.

[6] Z. Pan, Y. Chen, L. Kan and Y. Zhang. Parameter Estimation by Genetic Algorithms For Nonlinear Regression. In *Proceedings of the International Conference on Optimization Techniques and Applications*, pages 946–953. World Scientific, 1995.

[7] C. Rossi, M. Abderrahim and J. C. Díaz. Tracking Moving Optima Using Kalman-Based Predictions. *Evolutionary Computation*, 16(1): 1–30, MIT Press, 2008.

[8] G. A. F. Seber and C. J. Wild. *Nonlinear Regression*. Wiley & Sons, Inc, 2003.

[9] A. Simões and E. Costa. Evaluating Predictor's Accuracy in Evolutionary Algorithms for Dynamic Environments. In *Proceedings of GECCO-2009*. ACM, 2009.

[10] A. Simões and E. Costa. Improving Memory's Usage in Evolutionary Algorithms for Changing Environments. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2007)*, pages 276–283. IEEE Press, 2007.

[11] A. Simões and E. Costa. Using Linear Regression to Predict Changes in Evolutionary Algorithms dealing with Dynamic Environments. Technical Report TR 2007/005, ISSN 0874-338X, CISUC, 2007.

[12] A. Simões and E. Costa. Variable-size Memory Evolutionary Algorithm to Deal with Dynamic Environments. In M. Giacobini et al., editors, *Applications of Evolutionary Computing*, volume 4448 of *Lecture Notes in Computer Science*, pages 617–626. Springer, 2007.

[13] A. Simões and E. Costa. Evolutionary Algorithms for Dynamic Environments: Prediction using Linear Regression and Markov Chains. In *Parallel Problem Solving from Nature (PPSN X)*, pages 306–315. Springer, 2008.

[14] A. Simões and E. Costa. Prediction in Evolutionary Algorithms for Dynamic Environments Using Markov Chains and Nonlinear Regression. Technical Report TR 2009/01, ISSN 0874-338X, CISUC, 2009.

[15] P. D. Stroud. Kalman-extended Genetic Algorithm for Search in Nonstationary Environments with Noisy Fitness Evaluations. *IEEE Transactions on Evolutionary Computation*, 5(1): 66–77, 2001.

[16] J. van Hemert, C. Van Hoyweghen, E. Lukshandl and K. Verbeeck. A Futurist Approach to Dynamic Environments. In *GECCO EvoDOP Workshop*, pages 35–38, 2001.

[17] S. Yang. Explicit Memory Schemes for Evolutionary Algorithms in Dynamic Environments. In S. Yan, Y-S. Ong and Y. Jin, editors, *Evolutionary Computation in Dynamic and Uncertain Environments*, pages 3–28. Springer-Verlag, 2007.

Table 1: Prediction accuracy using linear regression and nonlinear regression in the $P1$ module of PredEA

Chg. period	Algorithm	Prediction Efficacy	Average Error	Average Err (abs)	Average Δ
periodic (r)	PredEA-LR	**100.00%**	0.00	0.00	5.00
	PredEA-NLR	**100.00%**	0.00	0.00	5.00
5-10-5	PredEA-LR	99.87%	-0.32	1.01	3.01
	PredEA-NLR	**100.00%**	-0.36	1.01	2.02
10-20-10	PredEA-LR	**99.73%**	-0.29	2.37	6.06
	PredEA-NLR	**99.73%**	-0.30	2.36	4.04
50-60-70	PredEA-LR	**99.66%**	-0.29	4.43	8.06
	PredEA-NLR	**99.66%**	-0.28	4.38	5.12
100-150-100	PredEA-LR	99.66%	-0.02	11.53	31.95
	PredEA-NLR	**100.00%**	-0.27	10.77	27.77
NL1	PredEA-LR	0.00%	-1885.52	1885.52	5.00
	PredEA-NLR	**98.45%**	-0.85	0.85	5.00
NL2	PredEA-LR	0.25%	778.56	778.56	592.53
	PredEA-NLR	**100.00%**	-0.39	0.62	2.07

Table 2: Global results for the dynamic bitmatching problem

Bitmatching Problem		Number of states									
		3		5		10		20		50	
r=10	noPredEA	89.68	90.63	85.99	86.53	80.74	81.97	75.27	76.53	69.74	70.89
	PredEA-LR	98.14	97.32	96.86	96.24	93.72	93.39	90.17	90.14	84.86	85.62
	PredEA-NLR	**98.24**	**97.37**	**96.89**	**96.27**	**93.76**	**93.49**	**90.18**	**90.18**	**84.89**	**85.64**
r=50	noPredEA	99.01	99.01	98.92	98.92	98.68	98.72	97.26	97.65	89.11	90.74
	PredEA-LR	99.86	99.85	99.79	99.80	99.64	99.58	99.27	99.22	98.20	98.03
	PredEA-NLR	**99.91**	**99.89**	**99.83**	**99.81**	**99.66**	**99.63**	**99.32**	**99.25**	**98.26**	**98.05**
r=100	noPredEA	99.50	99.50	99.44	99.45	99.31	99.33	99.12	98.82	94.44	95.42
	PredEA-LR	99.93	99.92	99.88	99.87	99.77	99.75	99.54	99.51	98.84	98.71
	PredEA-NLR	**99.94**	**99.93**	**99.89**	**99.88**	**99.78**	**99.76**	**99.55**	**99.52**	**98.86**	**98.72**
r=200	noPredEA	99.73	99.73	99.70	99.70	99.60	99.61	99.46	99.31	97.17	97.49
	PredEA-LR	**99.95**	99.94	99.91	99.90	**99.83**	99.81	99.65	99.63	99.13	99.06
	PredEA-NLR	**99.95**	**99.95**	**99.92**	**99.91**	**99.83**	**99.82**	**99.66**	**99.64**	**99.14**	**99.07**
5-10-5	noPredEA	85.56	89.94	85.59	86.53	80.97	81.48	74.05	75.73	67.99	69.04
	PredEA-LR	96.57	96.45	94.74	**94.04**	90.52	90.34	86.88	87.10	94.23	81.53
	PredEA-NLR	**96.89**	**96.76**	**94.90**	**94.04**	**90.58**	**90.45**	**86.89**	**86.26**	**94.59**	**81.85**
10-20-10	noPredEA	91.43	94.68	92.42	92.47	87.69	88.42	81.00	82.87	73.40	75.00
	PredEA-LR	97.79	98.31	97.67	96.40	95.14	94.22	92.80	92.32	87.22	86.52
	PredEA-NLR	**98.10**	**98.62**	**97.92**	**97.20**	**95.90**	**95.07**	**93.12**	**92.83**	**87.38**	**87.02**
50-60-70	noPredEA	99.15	99.15	99.06	99.07	98.83	98.86	98.15	97.88	90.78	92.34
	PredEA-LR	**99.89**	**99.87**	**99.82**	**99.80**	**99.63**	**99.61**	**99.25**	99.18	**98.09**	97.86
	PredEA-NLR	**99.89**	**99.87**	**99.82**	**99.80**	**99.63**	**99.61**	**99.25**	**99.19**	**98.09**	**97.89**
100-150-100	noPredEA	99.57	99.57	99.52	99.52	99.41	99.42	99.25	98.98	95.22	96.17
	PredEA-LR	**99.94**	**99.94**	99.91	99.90	99.81	99.80	99.62	**99.59**	99.02	98.91
	PredEA-NLR	**99.94**	**99.94**	**99.92**	**99.91**	**99.83**	**99.82**	**99.64**	**99.59**	**99.05**	**98.92**
NLR 1	noPredEA	98.95	98.94	98.73	98.77	98.38	98.49	97.02	97.41	91.60	92.83
	PredEA-LR	99.19	**98.68**	99.00	98.67	98.96	98.80	99.12	98.75	99.55	98.46
	PredEA-NLR	**99.74**	**99.68**	**99.64**	**99.61**	**99.48**	**99.48**	**99.25**	**99.22**	**99.58**	**98.53**
NLR 2	noPredEA	98.11	98.14	97.96	97.97	97.58	97.35	97.01	94.40	84.08	85.36
	PredEA-LR	98.12	98.15	97.96	97.97	97.57	97.35	97.01	94.22	84.10	85.43
	PredEA-NLR	**99.80**	**99.79**	**99.67**	**99.65**	**99.33**	**99.26**	**98.66**	**98.53**	**96.55**	**96.19**