Dynamic Multi-Objective Optimization with Evolutionary Algorithms: A Forward-Looking Approach

Iason Hatzakis and David Wallace
Massachusetts Institute of Technology
Room 3-458
77 Massachusetts Ave., Cambridge MA 02139
+1-617-258-6016
{iason, drwallac}@mit.edu

ABSTRACT

This work describes a forward-looking approach for the solution of dynamic (time-changing) problems using evolutionary algorithms. The main idea of the proposed method is to combine a forecasting technique with an evolutionary algorithm. The location, in variable space, of the optimal solution (or of the Pareto optimal set in multi-objective problems) is estimated using a forecasting method. Then, using this forecast, an anticipatory group of individuals is placed on and near the estimated location of the next optimum. This *prediction set* is used to seed the population when a change in the objective landscape arrives, aiming at a faster convergence to the new global optimum. The forecasting model is created using the sequence of prior optimum locations, from which an estimate for the next location is extrapolated. Conceptually this approach encompasses advantages of memory methods by making use of information available from previous time steps. Combined with a convergence/diversity balance mechanism it creates a robust algorithm for dynamic optimization. This strategy can be applied to single objective and multi-objective problems, however in this work it is tested on multi-objective problems. Initial results indicate that the approach improves algorithm performance, especially in problems where the frequency of objective change is high.

Categories and Subject Descriptors: I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search - Heuristic Methods.

General Terms: Algorithms, Design.

Keywords: Time-changing environment, dynamic problem, evolutionary algorithm, forecast, time series analysis.

1. INTRODUCTION

Evolutionary algorithms (EAs) in time-changing environments are asked to track a moving optimum as closely as possible. The performance of EAs in such dynamic environments has been studied mostly during the last two decades ([4], [5], [15], [23], [7], [10], [11] and [19]), and a plethora of performance-enhancement techniques has emerged. A large portion of the existing methods can be classified into two broad categories.

The first category deals with the control of the two basic functions of the algorithm's population in a dynamic environment: converging on the current global optimum and exploring the design space for the optimum's next location or for new optima as soon as the objective landscape changes. These two functions usually compete against each other and this competition can be viewed as a balance between convergence and diversity. Some of the techniques developed in the past direct the population toward convergence until a change arrives. Then they favor diversity in order to discover the new optimum. Such an example is the method of hypermutation found in Cobb [4] and Cobb and Grefenstette [5]. The balance between convergence and diversity has also been examined as a multi-objective problem, where the population diversity forms a second objective [13]. Other approaches, such as the Self-Organizing Scouts by Branke et al [9], separate the population into groups with specific functions of either tracking an optimum or exploring for new solutions.

The second broad category of approaches is concerned with exploiting past information (past fit solutions) which might again become useful as the problem evolves. The various memory methods are instances of such techniques ([8], [15] and [27]) that help the algorithm to avoid doing the same job more than once. The value of past information, such as the position of prior optima, has also been demonstrated by Branke et al. [12] in their discussion of changing environments.

The nature of the time-changing environment is another important aspect of dynamic optimization. Two basic categorizations of dynamic environments are the *frequency* and the *severity* of change ([11], [12]). If the severity of change is especially large, one instance of the problem will be completely unrelated to the next and it can be argued that a complete re-start of the algorithm is as efficient as any dynamic optimization methodology. On the other hand, dynamic environments may exhibit some form of structure in their evolution. More precisely, dynamic environments may be characterized by a smaller or larger amount of *predictability* in their temporal change pattern [12].

The approach proposed in this paper makes use of any predictability present in the objective's behavior to accelerate convergence and improve the performance of a dynamic evolutionary algorithm. More specifically, the past sequence of locations of the global optimum (or, realistically, of the best solution discovered by the algorithm during each time step) can be seen as a time series. There are many forecasting methods available that can be applied to provide an estimate of the next element in a time series given some past values, as described, for example, by Armstrong [2]. Some of these methods have emerged from statistics and econometrics ([1], [6], [18]), while others can be as simple as a polynomial extrapolation. Given an estimate of the next optimum's location, a group of individuals can be created on and near this estimate. This group anticipates the change in the landscape and, depending on the quality of the forecast, can potentially aid the discovery of the next optimum as it already lies near it. Bosman [3] states the importance of learning and of preparing the population in anticipation of the objective's future behavior. This method will be called *feed-forward prediction strategy* (FPS). In principle the FPS can be used for both single- and multi-objective problems. In this work it will be tested on a two-objective problem.

In Section 2 the feed-forward prediction strategy is described and various practical issues that emerge are commented on. In Section 3 the evolutionary algorithm used for the study is defined, and in Section 4 details are provided on the current implementation of the FPS. Results from a two-objective test problem are given in Section 5.

2. FEED-FORWARD PREDICTION STRATEGY

2.1 General Description

The strategy proposed in this work consists of using the past history of the optimum's path over time to predict the optimum's location in the next time step. The prediction for the optimum's next location is used to create a set of individuals (consisting, in the simplest case, of a single individual that lies on the prediction coordinates) which is called the *prediction set*. As soon as the next time step arrives and the objective function changes, the prediction set is inserted in the algorithm's population. A simplified illustration of this concept for a two dimensional design vector is shown in Figure 1. If the prediction is successful then the individuals in the prediction set are close to the next optimum in the design space. Hence in this case the prediction set aids the population to discover the new optimal solution quickly.

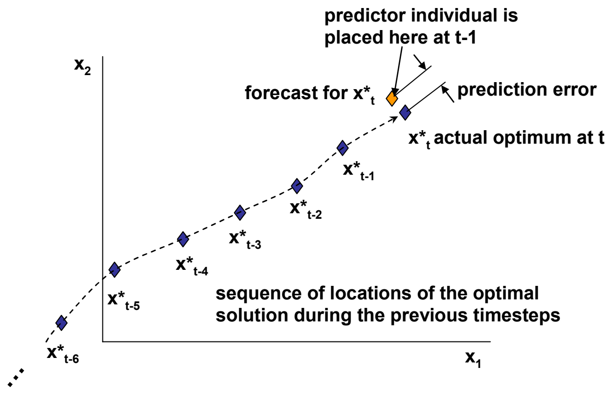
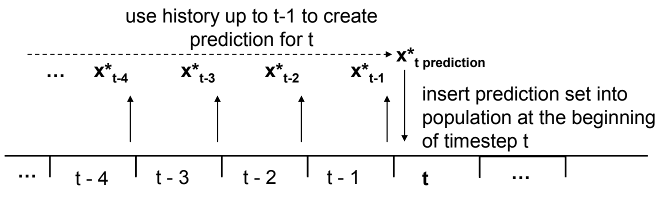

predictor individual is placed here at t-1
x_2
forecast for x^*_t
prediction error
x^*_t actual optimum at t
x^*_{t-1}
x^*_{t-2}
x^*_{t-3}
x^*_{t-4}
x^*_{t-5}
x^*_{t-6}
sequence of locations of the optimal solution during the previous timesteps
x_1

Figure 1. The sequence of past optima (up to t-1) is used to create a prediction for the next time step t.

The use of a prediction set in order to discover the new optimum quickly and efficiently is not intended to be completely self-sufficient as a dynamic optimization technique. This is because the prediction might not be successful - there might not be predictability in the dynamic behavior of the objective, or the pattern might not be identifiable by the forecasting method. In such a case, other dynamic optimization techniques are required. As described next, our approach uses a balance between convergence and diversity along with the feed-forward prediction strategy in order to aid discovery of the new optimum in the case when the forecast is unsuccessful.

Our proposed method is intended for dynamic problems of a discrete nature, where the objective changes in a non-infinitesimal way and the optimization engine has a fixed number of function evaluations available at each time step. The feed-forward prediction strategy is outlined in the following pseudo-code, and in Figure 2:

```
At the end of time step t-1:
1. Using the time series of the current and
   past locations of the optimal solution {$x^*_{t-1}$, $x^*_{t-2}$, ...} and a forecasting method,
   create a prediction for the location of the
   next optimal solution $x^*_t$.
2. Using the prediction a group of individuals
   (the prediction set) is created.
At time step t:
3. The prediction set is inserted into the
   population.
4. The optimization algorithm runs for the
   available function evaluations to discover
   the new optimum $x^*_t$.
5. At the end of the time step the best
   discovered solution is stored as the
   optimum at time t, $x^*_t$, and the time series
   is updated.
6. Return to 1.
```


use history up to t-1 to create prediction for t
$x^*_{t\ prediction}$
... x^*_{t-4} x^*_{t-3} x^*_{t-2} x^*_{t-1}
insert prediction set into population at the beginning of timestep t
... t - 4 t - 3 t - 2 t - 1 t ...

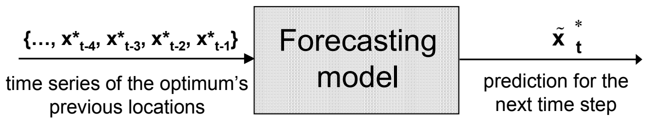

{..., x^*_{t-4}, x^*_{t-3}, x^*_{t-2}, x^*_{t-1}}
time series of the optimum's previous locations
Forecasting model
$\tilde{x}^*_t$
prediction for the next time step

Figure 2. Outline of the feed-forward prediction strategy

As previously mentioned, a convergence/diversity balance technique is used so that the new optimum can be discovered even if the prediction is not successful. The total population at the beginning of a time step (step 4 in the previous pseudo-code) is composed of three parts (see Figure 3): the non-dominated *front*, whose function is to converge to the current solution, the dominated set (called the *cruft*), whose function is to preserve diversity in order to be able to search and discover new optima, and the *prediction set*, which places a team of individuals in the neighborhood of the next optimum in order to achieve faster discovery and convergence. The prediction set handles the predictable portion of the objective's change pattern, while the cruft handles any unpredictable change and helps discover the new optimum even if the prediction contains a large error. More detail on how convergence and diversity are affected by the front and cruft sub-populations follows in section 3.

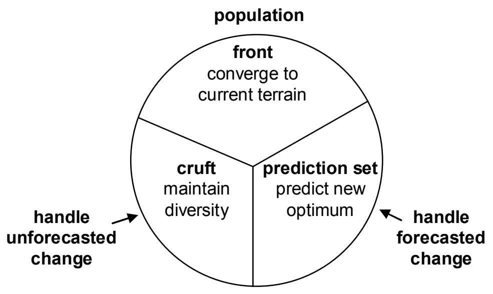


Figure 3. Population subgroups.

Hence the feed-forward prediction strategy is intended to have three basic qualities: First, the ability to respond to changes in the landscape and explore for new solutions; Second, the ability to take advantage of any structure and predictability in the objective's temporal change pattern in an active way by anticipating the change and moving towards the new optimum; and Third, the ability to exploit past information by using the best solution's history to train the forecasting model in order to accelerate convergence. These qualities may combine the advantages of both convergence/diversity balance and memory methods with a feed-forward approach, making use of all the available information and the work done by the algorithm up to the present point.

This methodology can perform well in cases where the landscape's temporal change pattern exhibits some degree of predictability. It is felt that this requirement is not overly limiting since many dynamic problems which are completely unpredictable are also prone to be as efficiently solved with a simple re-start at every time step as with a dynamic optimization algorithm. When there is little or no predictability in the temporal change pattern, the diversity provided by the cruft allows the discovery of the new optimum. The prediction set normally contains a very small number of individuals (usually less than 5% of the total population) and thus the expense it induces in terms of function evaluations is small compared to the overall population.

2.2 Multi-objective problems

In a single objective problem, we are interested in tracking a single global optimum or a set of local optima. In a multi-objective problem the solution is the non-dominated Pareto front which, in the two-objective case, is a curve described by an infinite number of points. The question that arises in this case is: how to track and predict the Pareto front?

There are a number of ways to try to address this question. A straightforward approach (followed in this work) is to select some points (vertices) on the Pareto front and track them. In the results that follow the two anchor points ($\min(f_1)$ and $\min(f_2)$) of the Pareto front were used as vertices and were tracked and predicted as next-step optima. Other methods could also be used. One could include more than two vertices of the Pareto front in the prediction set. One might also fit an analytic curve which describes the Pareto optimal set (the locus of the Pareto front in variable space) and subsequently forecast changes in the parameter values of this curve, instead of using specific points.

2.3 The prediction set

The simplest approach in creating the prediction set from the forecast is to place a single individual on each forecasted location. The prediction set in the results that follow was created this way. More elaborate methods could be used to provide a comprehensive coverage of the predicted optimum's area. For example, a hypercube of individuals could be created around the predicted location of the next optimum. Information on the error and the confidence margins of the forecast could be used to dimension the hypercube so that there is a high likelihood it includes the actual optimum.

2.4 Two sources of prediction error

Ideally the forecasting model would predict the exact location of the global optimum (or of the Pareto optimal set) for the next time step. The actual prediction, however, will likely have an amount of error in it. There are two main sources for this error.

2.4.1 Accuracy of the optimal solution's history

The input of the forecasting model is a time series of the optimal solution's location during the previous time steps. The actual optimum is of course unknown – the available data is the best discovered solution at each time step. If the evolutionary algorithm has not converged properly, then the time series of the best discovered solutions does not coincide with the time series of the true optima, which induces error in the input of the forecasting model.

2.4.2 Accuracy of the forecasting model

Even if the input data is accurate, there is a possibility that the forecasting model will produce an inaccurate prediction. This depends on the nature of the problem and the type and quality of the forecasting model. For example, if the problem is linear and deterministic (i.e. the global optimum traces a straight line segment in the variable space) and we are using a first-order polynomial extrapolation as a forecasting model, then there will be no error in the forecast of the optimum's location in the next time step. If, on the other hand, the environment's temporal change pattern is stochastic then the forecast will most possibly not coincide with the next time step's optimum due to the randomness which comes into defining the optimum's location.

2.5 Forecasting model

In order to keep the method as widely applicable as possible, one needs to use a forecasting model which makes no assumptions on the nature of the objective's temporal change pattern (such assumptions could regard for example linearity or periodicity). One also needs to assume that the algorithm is blind and the only information available to the forecasting model is the location of the previous time steps' best discovered solutions.

In the current implementation, stochastic time series models are used. These models result from the time series analysis and forecasting methods which can be found in statistics and econometrics [2]. They include Autoregressive (AR) and Moving Average (MA) techniques (often termed *Box-Jenkins* methods, [6]) and a number of their variants (Autoregressive Integrated Moving Average models (ARIMA), Vector (multivariate) Autoregressive models (VAR) etc.). Time series methods have been developed extensively [17]. They are also intended for use with random processes and hence, in this context, they can be applied to stochastic optimization problems. A disadvantage of time series methods is that they require certain conditions to be met regarding the statistical characteristics of the data – for example, autoregressive models require the data to be mean and covariance-stationary ([1], [17]). However it is possible to identify and treat the cases when these conditions do not hold ([17], [18]).

In addition to econometric methods, other possible forecasting model candidates range from a simple polynomial extrapolation to an artificial neural network. Any mathematical process which can produce a one-step-ahead estimate of the best solution's location could serve as a forecasting model.

3. EVOLUTIONARY ALGORITHM

The evolutionary algorithm used in this work was developed by Leyland [21] and the first author. It has been based on the Queuing Multi-Objective Optimizer (QMOO) created by Leyland [21]. QMOO has been extensively tested for static single- and multi-objective problems and is used within the Distributed Object-Based Modeling Environment (DOME) [26]. The algorithm used here is called D-QMOO (Dynamic QMOO), a development of QMOO that includes dynamic optimization methods.

D-QMOO is an elitist, steady-state[1], clustering, multi-objective evolutionary algorithm. In each cluster, the population contains two sub-groups (recall Figure 3). The front, which contains non-dominated individuals of rank 1, and the dominated set (cruft) which contains dominated individuals of any rank greater than one. A fixed fraction of the population is dedicated to each of the two sub groups. If the feed-forward prediction strategy is used, a third subgroup (the prediction set, recall Figure 3) is also inserted in the population when a change in the objective function arrives. The individuals in the prediction set (*predictor individuals*) are subsequently absorbed by the front or the cruft depending on their fitness.

Parent selection is random using a uniform distribution and drawing from the whole population (both front and cruft). Elitism is ensured by the fact that only non-dominated individuals can enter the front. When the front reaches its size limits, elimination of individuals is done in such a way as to preserve the maximum dominated volume. Any dominated individual may enter the cruft. Elimination of individuals in the cruft is performed in two different ways: either by the individual's age (oldest individuals die) or by crowding in variable space (individuals in more crowded areas die).

D-QMOO manages the balance between convergence and diversity in two ways: through the existence of the front and the cruft subgroups, and through the selection of the thinning method used in the cruft (age or crowding). The existence of the cruft and the selection of its thinning method ensures that a non-elitist, diverse group always remains in the population and can react to changes in the objective, helping the algorithm discover the new optimal set. A number of different rules can be used for thinning the cruft, such as an open-loop control rule which favors thinning by crowding when the objective has just changed, and then gradually reverts to thinning by age as long as the objective remains constant. In this work the thinning method is selected by a simple flip-coin (Bernoulli) experiment with a fixed probability every time a cruft individual has to die.

Using the convergence-diversity balance, D-QMOO can handle dynamic optimization problems with or without the feed-forward prediction strategy presented in this paper. The feed-forward prediction strategy is aimed at increasing the optimization performance of the algorithm – conceptually, it could be used with any dynamic evolutionary optimizer (steady-state or generational). It can be claimed that a desirable characteristic for an algorithm employing the FPS is elitism, since the prediction set is a hopefully fit but small group and elitism can help it survive.

4. IMPLEMENTATION

Details of the current implementation of the feed-forward prediction strategy are provided in this section. This implementation produced the results presented in the following section. Two different variants of the FPS are used and their performance will be compared to that of D-QMOO without any feed-forward strategy.

4.1 Evolutionary algorithm

D-QMOO is used as described in section 3. The population consists of 100 individuals of which 70 belong to the front and 30 to the cruft. Cruft thinning is done using fixed equal probabilities (0.50 – 0.50) of selecting thinning by age and thinning by crowding.

As currently implemented, the algorithm knows when a change in the objective arrives. This may or may not be the case in a real-world problem. Either way, this is a significant assumption. Change detection in dynamic environments is itself an important topic which has been addressed by various researchers in the past. They have proposed techniques such as observing the algorithm's performance (see for example Cobb's work [4]), or explicitly monitoring for landscape variations (like Morrison's sentinel EA [22] does). However, this issue will not be treated presently.

4.2 Forecasting model

An autoregressive (AR) model created by Schneider and Neumaier [24] is used as a forecasting method. Two different options for the prediction of an individual's location are used:

- The whole design vector is treated as a vector time series and a multivariate vector autoregressive (VAR) model is used for forecasting. In this case, the forecast $\tilde{\mathbf{x}}_t$ for the t – time step's design vector $\mathbf{x}_t$ is [24]:

  (1) $$\tilde{\mathbf{x}}_t = \mathbf{w} + \sum_{i=1}^{p} \mathbf{A}_i \cdot \mathbf{x}_{t-i}$$

  where $\mathbf{x}$ is the $n\times1$ design vector, $\mathbf{A}_i$ are the $n\times n$ autoregressive coefficient matrices and p is the order of the autoregressive model (selected by Schwarz's Bayesian Criterion [24]). The $n\times1$ vector $\mathbf{w}$ is an intercept term which allows for a non-zero mean of the time series. The algorithm using this method is called D-QMOO/VAR.

[1] steady state evolutionary algorithms have been found by Vavak and Fogarty [25] to perform well against generational algorithms in dynamic optimization problems

- Each design variable is treated as a single time series and a univariate autoregressive model is applied for the forecasting of each variable $x_{j,t}$ of the design vector separately:

  (2) $\tilde{x}_{j,t} = w_j + \sum_{i=1}^{p} a_{j,i} \cdot x_{j,t-i}$, for $j = 1,...,n$

  As before but in scalar form, $a_{j,i}$ are the autoregressive coefficients and w_j is the intercept term. The algorithm using this method is termed D-QMOO/AR.

Hence, three versions of the dynamic optimization algorithm are tested and compared: the multivariate D-QMOO/VAR, the univariate D-QMOO/AR, and the algorithm without any feed-forward prediction strategy (D-QMOO).

Initially the algorithm is run for a fixed number of time steps (training period). In Section 5's experiments this period is 100 time steps. The sequence of best discovered solutions at each time step is collected into a time series and the AR model is fitted. Subsequently at the end of each time step the best discovered solution is added to the time series, and the AR model is used to forecast the location of the optimal solution (predictor individual) for the next time step.

4.3 Prediction set

The method is tested on two-objective problems and a prediction set that only includes the two anchor points of the Pareto front is used. The prediction set consists of the forecasts for the two anchor points. Hence two time series are kept, one for each anchor point. A separate AR model is fitted onto each of the time series. These models are used in order to get a prediction for the location of each anchor point in the next time step. Two corresponding individuals are created forming the prediction set. The prediction set is inserted into the starting population as soon as the objective changes onto the next time step. If the prediction is successful, then the prediction set is close to the actual anchor points, helping the population converge faster to the new Pareto front.

5. RESULTS

In this section results from the application of the feed-forward prediction strategy are presented. Dynamic optimization problems to which this strategy would appeal most are problems in which the location of the Pareto optimal set changes in time. The FDA1 problem from the dynamic multi-objective test suite proposed by Farina et al [14] is selected.

The results from using the FPS are very encouraging, especially in the case of the D-QMOO/AR univariate version since it significantly improves the solution performance in high change frequencies.

5.1 FDA1 test problem

In this two-objective problem, the Pareto optimal set is harmonically varying in time between two extrema for all but one of the variables. The FDA1 problem is defined as follows [14]:

(3)
$$\text{minimize } \mathbf{f}(f_1(\mathbf{x}_\text{I}, t), f_2(\mathbf{x}, t))$$
$$\text{where } f_1(\mathbf{x}_\text{I}, t) = x_1$$
$$f_2(\mathbf{x}, t) = g(\mathbf{x}_\text{II}, t) \cdot h(\mathbf{x}_\text{III}, f_1(\mathbf{x}_\text{I}, t), g(\mathbf{x}_\text{II}, t))$$
$$\text{and } \mathbf{x} = \left[\mathbf{x}_\text{I} \ \ \mathbf{x}_\text{II}\right]^T = \left[x_1 \ x_2 \ \cdots \ x_n\right]^T \text{ is the design vector}$$

The g and h functions are defined:

(4)
$$g(\mathbf{x}_{II}) = 1 + \sum_{\mathbf{x} \in \mathbf{x}_\text{II}} (x_i - G(t))^2$$
$$h(f_1, g) = 1 - \sqrt{f_1 / g}$$
$$G(t) = \sin(0.5\pi t), \ t = \frac{1}{n_t} \left\lfloor \frac{\tau}{\tau_T} \right\rfloor$$
$$\mathbf{x}_I = (x_1) \in [0,1], \ \mathbf{x}_{II} = (x_2, ..., x_n) \in [-1,1]$$

In this definition, τ is the current number of function evaluations, τ_T is the number of function evaluations for which the time t remains fixed (and the objective remains constant), and n_t is the number of distinct time steps in one time unit. The Pareto optimal front is analytically solved to be:

(5)
$$f_2 = 1 - \sqrt{f_1}$$

And the Pareto optimal set is:

(6)
$$x_1 \in [0,1]$$
$$x_{2,...,n} = G(t) = \sin(0.5\pi t)$$

Hence, according to the analytic solution of the problem, x_1 spans [0,1] evenly in order to cover the Pareto optimal set and the rest of the design vector oscillates harmonically in time between -1 and 1.

The design vector has a dimension n = 10, and a time discretization of n_T= 10 time steps per time unit is employed. The n_T parameter controls the problem change severity, since it dictates the number of time steps in one full cycle (which corresponds to four time units). The change period is τ_t = 500 evaluations per time step. Severity is constant throughout this work in order to observe the effect of changes in the number of evaluations per timestep. All results in this section are averages of 20 runs for each case. The averaged results of Table 1 show the feed-forward prediction strategy to have a positive effect. The effect is stronger when using the univariate model, where a separate autoregressive model is used to predict the next time step's value for every variable in the design vector: using the D-QMOO/AR algorithm brings a reduction of 31% in the Pareto front error and almost 50% in the design vector error when compared to no feed-forward strategy. This signifies a very positive effect from feed-forward prediction strategy. The multivariate model (D-QMOO/VAR) also has a positive but much smaller effect: about 2.5% reduction in the Pareto front and around 3.4% reduction in the design vector error.

A possible reason for the better performance of the univariate model is that each design variable's forecast is isolated from the other variables. In contrast, in the multivariate model each variable is affected by the whole design vector and hence an error in one design variable can propagate to the others. This is a problem-specific result. A forecasting model which takes into account the linkage between design variables might perform better in problems with strongly coupled variables.

Table 1: Objective (Pareto front convergence) and design vector error. The objective (Pareto) and the design vector errors are calculated as described by Farina et al [24], expressing the deviation of the current population from the actual solution. $\overline{e}_f$ and $\overline{e}_x$ denote time averages of the errors during each run, and their mean and st.dev. over 20 runs of 300,000 evaluations is shown.

	objective error		design vector error	
algorithm	$\overline{e}_f$ mean	$\overline{e}_f$ st. dev.	$\overline{e}_x$ mean	$\overline{e}_x$ st. dev.
D-QMOO/AR (univariate)	0.02984	$9.44\ 10^{-4}$	0.00298	$1.62\ 10^{-4}$
D-QMOO/VAR (multivariate)	0.04231	$7.34\ 10^{-4}$	0.00573	$1.70\ 10^{-4}$
D-QMOO (no predictor)	0.04336	$6.94\ 10^{-4}$	0.00593	$1.41\ 10^{-4}$

In the next set of numerical experiments, the frequency of change for the objective function is varied. This parameter controls the number of objective function evaluations between changes in the objective (number of evaluations per time step) and affects the solution quality in a dynamic optimization problem since it sets the amount of time available to the algorithm for convergence. Hence, a measure of merit in discrete dynamic environments is how closely the algorithm converges to the true Pareto front at each time step for a given change frequency or, conversely, the maximum change frequency for which the algorithm can provide a given solution accuracy. This solution accuracy can be quantified by the Pareto and design vector errors. The experiments presented here show that the feed-forward prediction strategy can especially augment the performance of an evolutionary algorithm in a dynamic environment at high change frequencies (low change periods), where the algorithm would otherwise perform poorly.

In the following results the change period is varied from 500 to 30,000 evaluations per time step. Results are averages over 20 runs. The error at each time instant is measured at the end of the time step just before the objective changes. The positive effect of using the feed-forward prediction strategy is apparent if one looks at Figure 4. As soon as the predictor is switched on at t = 10, the mean objective error drops by almost 37%, from 43.1 10^{-3} to 27.2 10^{-3}.

The effect of a decreasing change frequency can be seen in Figures 5 and 6 where the change period is increased to 2000 and 5000 respectively. The benefit from using the feed-forward strategy attenuates as the change period increases. In Figure 6 (5000 evaluations per time-step) the effect of the predictor is almost imperceptible. The starting error is smaller as the period increases since the algorithm has more function evaluations available and hence a better chance to converge to the Pareto optimal set during each time step, independent of the feed-forward prediction. However for higher frequencies the feed-forward prediction strategy has an obvious beneficial effect, especially when using the univariate autoregressive model (D-QMOO/AR algorithm). This is also apparent if we examine the average objective and design vector error in Figure 7 and Figure 8. In low periods D-QMOO/VAR performs better than D-QMOO, and the univariate D-QMOO/AR performs significantly better than the first two. As the period increases, the performance of the three algorithms converges to the same level.

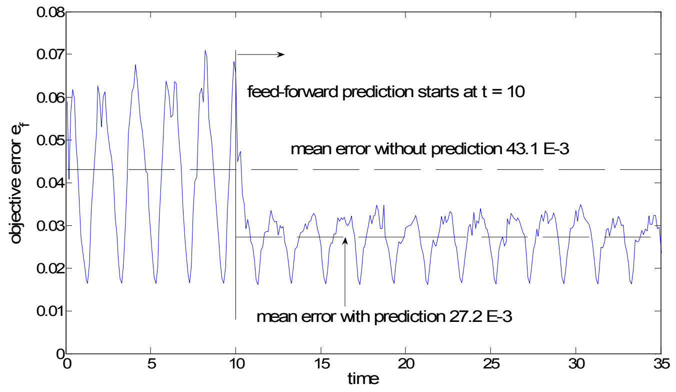


Figure 4. Objective error during the run. Change period 500 evaluations per time step. D-QMOO/AR algorithm. The positive effect of FPS in high frequency is apparent.

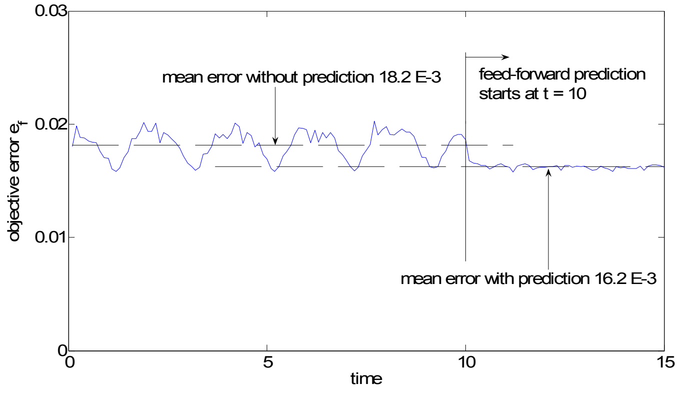


Figure 5. Objective error during the run. Change period 2000 evaluations per time step. D-QMOO/AR algorithm.

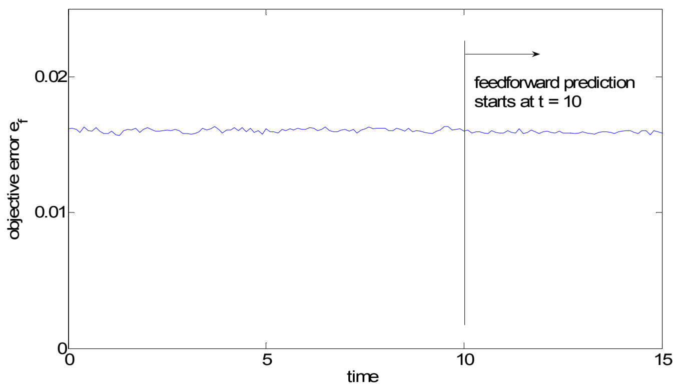


Figure 6. Objective error during the run. Change period 5000 evaluations per time step. D-QMOO/AR algorithm. The prediction's effect in this lower frequency is negligible.

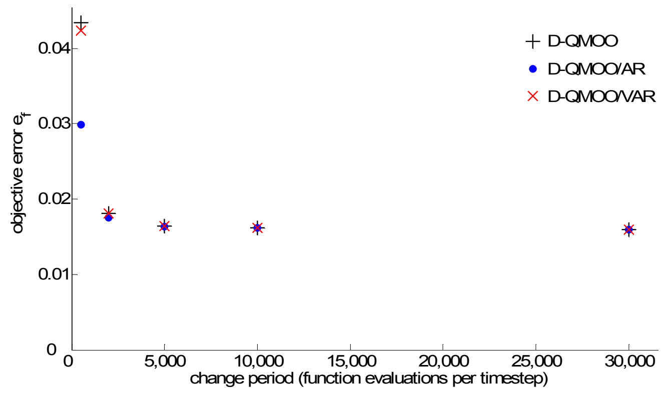


Figure 7. Objective (Pareto) error with change period. Initially D-QMOO/AR has a significantly smaller error than D-QMOO. The performance of the three algorithms converges as the period increases and the effect of the FPS attenuates.

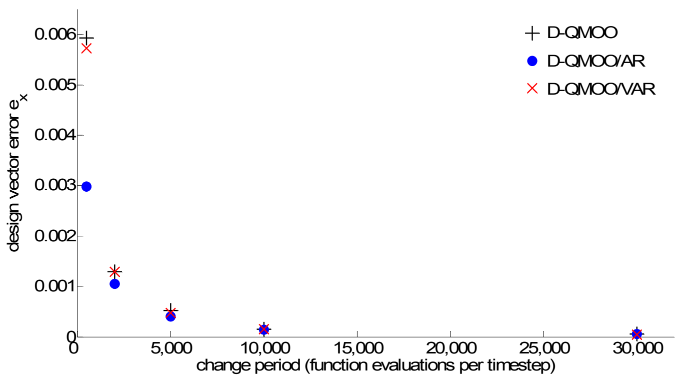


Figure 8. Design vector error with change period.

An illustration of the function of the prediction set can be seen in the snapshots of Figure 9. At the beginning of the time step, the bulk of the population is still around the previous time step's locus of the Pareto optimal set. The two predictor individuals however already lie near the actual solution of the current time step. They have been created at the end of the previous time step through the autoregressive model's forecast, and their function is to 'show the way' for the rest of the population towards the new solution. At the end of the time step, the front individuals have almost all converged onto the Pareto optimal set.

6. CONCLUSION AND FUTURE DIRECTIONS

The feed-forward prediction strategy is a conceptually interesting approach to address time-changing problems with evolutionary algorithms, since it adopts a forward-looking attitude in which the optimization algorithm exploits past information and prepares for the change before it arrives instead of simply reacting to it. Initial results from the implementation of the method are promising. The feed-forward prediction strategy improves the solution accuracy, especially in the critical cases where the frequency of change is high and the algorithm has otherwise little time to converge to the Pareto front.

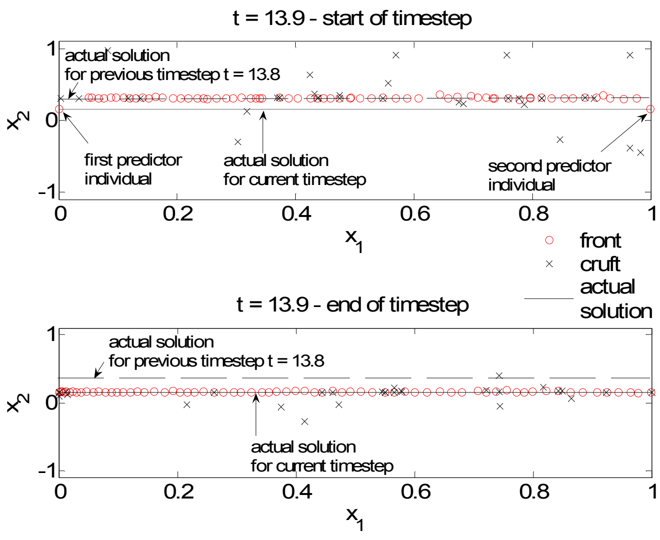


Figure 9. In the top figure, the time has just advanced to 13.9 and we are at the start of the new (current) time step. The two predictor individuals, one for each anchor point of the Pareto plot, already lie near the current actual solution (shown as a continuous line), while the rest of the population is still around the previous time step's Pareto optimal set (shown as a dashed line). At the end of the time step (bottom figure), the rest of the front individuals have followed the prediction set and converged onto the current Pareto optimal set. (D-QMOO/AR, 5000 evaluations per time step)

The methodology and implementation presented here are only a subset of a large scope of possible forms that the feed-forward prediction strategy can take. Further exploration of this methodology is needed for many aspects. Evaluation with other single- and multi-objective test problems (e.g. [8], [20]) and with practical applications would be beneficial. The use of different kinds of forecasting models can be examined along with the type of dynamic problems they are suited to. Apart from autoregressive models one may use polynomial extrapolation, neural networks, Bayesian models and other forecasting methods. It is rational to assume that statistical extrapolation models such as the ones used in this work will be especially suited to handle stochastic optimization problems. The topology of the prediction set individuals is another interesting topic since the prediction set can be given complex forms, such as a hypercube around the prediction coordinates which may help the algorithm discover the new solution faster. This issue is the subject of ongoing work by the authors. Finally, in multi-objective problems, an important question is which vertices of the Pareto front to track and include in the prediction set – the approach of using the two anchor points seemed to perform well in this work, but this aspect has not been explored thoroughly yet.

7. ACKNOWLEDGMENTS

The authors would like to thank Dr Geoff Leyland, whose constant help and advice has been vital for the continuation of this research, and to Dr Marco Farina and Prof. Kalyanmoy Deb for their help in using the FDA test problems. We would also like to thank the reviewers for their insightful comments which helped to enrich this paper. This work has been funded by the Center for

Innovation in Product Development (CIPD) of MIT, through the Product Development Activities Fund. This support is gratefully acknowledged.

8. REFERENCES

[1] Akaike, H., and Nakagawa, T. 1972. *Statistical Analysis and Control of Dynamic Systems.* KTK Scientific Publishers, Tokyo.

[2] Armstrong, J. S., ed. 2001. *Principles of Forecasting: A Handbook for Researchers and Practitioners*. Kluwer, Norwell, MA.

[3] Bosman, P. A. N. 2005. Learning, Anticipation and Time-Deception in Evolutionary Online Dynamic Optimization. In *GECCO-2005 Workshop on Evolutionary Algorithms for Dynamic Optimization*, Washington DC.

[4] Cobb, H. 1990. An Investigation into the Use of Hypermutation as an Adaptive Operator in Genetic Algorithms Having Continuous, Time-Dependent Nonstationary Environments. *NRL Memorandum Report 6760*.

[5] Cobb, H. G. and Grefenstette, J. J. 1993. Genetic Algorithms for Tracking Changing Environments. In *Proceedings of the 5th international Conference on Genetic Algorithms* S. Forrest, Ed. Morgan Kaufmann Publishers, San Francisco, CA, 523-530.

[6] Box, G., Jenkins, G., and Reinsel, G. 1994. *Time Series Analysis – Forecasting and Control.* Prentice Hall, New Jersey, NJ.

[7] Branke, J. 1999. *Evolutionary Approaches to Dynamic Optimization Problems – A Survey*, Technical Report 387, Institute AIFB, University of Karlsruhe.

[8] Branke, J. 1999. Memory Enhanced Evolutionary Algorithms for Changing Optimization Problems. In *Congress on Evolutionary Computation CEC99*, 3, pp. 1875-1882, IEEE.

[9] Branke, J., Kaussler, T., Schmidt, C. and Schmeck, H. 2000. A Multi-Population Approach to Dynamic Optimization Problems. In *Adaptive Computing in Design and Manufacturing 2000*, pp. 299-308, Springer.

[10] Branke, J. 2001. Evolutionary Approaches to Dynamic Optimization Problems – Updated Survey. In *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, pages 27–30.

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

[12] Branke, J., Salihoğlu, E., and Uyar, Ş. 2005. Towards an analysis of dynamic environments. In *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation* (Washington DC, USA, June 25 - 29, 2005). H. Beyer, Ed. GECCO '05. ACM Press, New York, NY, 1433-1440.

[13] Bui, L., Branke, J., Abbass, H. 2004. *Multiobjective Optimization for Dynamic Environments*. Artificial Life and Adaptive Robotics Laboratory Technical Report TR-ALAR-200504007, Canberra.

[14] Farina, M., Deb, K. and Amato, P. 2004. Dynamic Multiobjective Optimization Problems: Test Cases, Approximations and Applications. *IEEE Trans. Evol. Comp. vol. 8(5).*

[15] Goldberg, D. E. and Smith, R. E. 1987. Nonstationary function optimization using genetic algorithm with dominance and diploidy. In *Proceedings of the Second international Conference on Genetic Algorithms on Genetic Algorithms and their Application* (Cambridge, Massachusetts, United States). J. J. Grefenstette, Ed. Lawrence Erlbaum Associates, Mahwah, NJ, 59-68.

[16] Grefenstette, J. 1992. Genetic algorithms for changing environments*. Parallel Problem Solving from Nature*, 2. pp. 137 - 144. (Reinhard Manner and Bernard Manderick, Eds.). Amsterdam.

[17] Hamilton, J. 1994. *Time Series Analysis.* Princeton University Press, Princeton, NJ.

[18] Harris, R., and Sollis, R. 2003.*Applied Time Series Modeling and Forecasting.* Wiley.

[19] Jin, Y. and Branke, J. 2005. Evolutionary Optimization in Uncertain Environments—A Survey. *IEEE Trans. Evol. Comp.*, Vol. 9, No. 3, June 2005.

[20] Jin, Y. and Sendhoff, B. 2004. Constructing Dynamic Optimization Test Problems Using the Multi-objective Optimization Concept. In *Lecture Notes in Computer Science*, 3005, Jan 2004, 525 – 536.

[21] Leyland, G. 2002. *Multi-Objective Optimization Applied to Industrial Energy Problems.* PhD Thesis, EPFL, Lausanne.

[22] Morrison, R. W. 2004. *Designing Evolutionary Algorithms for Dynamic Environments.* Springer-Verlag, Berlin.

[23] Ronnewinkel, C., Wilke, C. O., and Martinetz, T. 2001. Genetic algorithms in time-dependent environments. In *theoretical Aspects of Evolutionary Computing* Natural Computing Series. Springer-Verlag, London, 261-285.

[24] Schneider, T. and Neumaier, A. 2001. ARFIT – A Matlab package for the estimation of parameters and eigenmodes of multivariate autoregressive models, *ACM Trans. Math. Soft.,* Vol. 27(1), March 2001, pp. 58-65.

[25] Vavak, F. and Fogarty, T. C. 1996. Comparison of Steady State and Generational Genetic Algorithms for Use in Nonstationary Environments. In *Proceedings of the Society for the Study of Artificial Intelligence and Simulation of Behavior; workshop on Evolutionary Computation '96*, pages 301-307, University of Sussex.

[26] Wronski, J. 2005. *A design tool architecture for the rapid evaluation of product design tradeoffs in an Internet-based system modeling environment*. SM Thesis, MIT, Cambridge, MA.

[27] Yamasaki, K. 2001. Dynamic Pareto Optimum GA against the changing environments. *In Proc. Genetic Evolutionary Computation Conf. Workshop Program*, San Francisco, CA, July 2001, pp. 47–50.