10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference
30 August - 1 September 2004, Albany, New York

AIAA 2004-4379

Evolving Intervening Variables for Response Surface Approximations

András Sóbester*, Prasanth B. Nair†
Andy J. Keane‡
University of Southampton, Southampton, Hampshire, SO17 1BJ, UK

Genetic Programming (GP) is a powerful string processing technique based on the Darwinian paradigm of natural selection. Although initially conceived with the more general aim of automatically producing computer code for complex tasks, it can also be used to evolve symbolic expressions, provided that we have a fitness criterion that measures the quality of an expression. In this paper we present a GP approach for generating functions in closed analytic form that map the input space of a complex function approximation problem into one where the output is more amenable to linear regression. In other words, intervening variables are evolved in each dimension, such that the final approximation model has good generalization properties and at the same time, due to its linearity, can easily be incorporated into further calculations. We employ least squares and cross-validation error measures to derive the fitness function that drives the evolutionary process. Results are presented for a one-dimensional test problem to illustrate some of the proposed ideas – this is followed by a more thorough empirical study, including multi-dimensional approximations and an engineering design problem.

Nomenclature

$\mathbf{x}$	=	vector of design variables
k	=	number of design variables (problem dimensionality)
n	=	number of training points
y	=	response
$\hat{y}$	=	approximated response
x_i	=	i^{th} element of $\mathbf{x}$
$\boldsymbol{\xi}$	=	vector of intervening variables
$\mathbf{z}$	=	vector of design variables in the transformed space
$\phi(.)$	=	basis function
$\mathbf{c}$	=	basis function centre (k-vector)
α_i	=	the weights of the basis function predictor

I. Introduction

RESPONSE surface approximation techniques are widely used in modern engineering design practice to model relationships in observational data of the form $(\mathbf{x}_i, y(\mathbf{x}_i)), i = 1 \ldots n$. Here, $\mathbf{x}_i$ denotes a k-dimensional input vector of design variables and y is the response, typically the output of an expensive computer simulation. A number of techniques exist in the literature for constructing response surface approximations: linear and polynomial regression,[1] radial basis functions,[2] Kriging,[3] support vector machines,[4] etc. The approximate model can subsequently be used as a computationally cheap surrogate *in lieu* of the original computer simulation to expedite design optimization and

*Research Fellow, Computational Engineering and Design Group, AIAA Member.
†Senior Research Fellow, Computational Engineering and Design Group, AIAA Member.
‡Professor of Computational Engineering, Chair of Computational Engineering and Design Group.

uncertainty analysis studies.[5–7] Here we focus our attention on linear models, which have important applications in several areas. For example, they are ideal for uncertainty analysis, where the analytical modeling of the propagation of probability distributions is often intractable when a nonlinear model is fitted to the data.

Of course, the major drawback of linear response surface models is that they tend to generalize badly over highly nonlinear training data. To overcome this, the present work revisits an idea originally proposed in the 1970s by Schmidt and co-workers[8] in the context of structural optimization: that of intervening variables as a means of enhancing the accuracy of response surface models.* At the basis of the approach adopted here stands the replacement of the original design variable vector $\mathbf{x}$ with the intervening variable vector $\boldsymbol{\xi}$, such that the function to be approximated, which is originally nonlinear in $\mathbf{x}$, turns out to be approximately linear in $\boldsymbol{\xi}$.

On the following pages we will argue that Genetic Programming (GP) is a flexible, effective means of "discovering" such intervening variables. The fundamentals of this idea are set out in the next section, followed by a simple one-dimensional example that illustrates them (Section III). More complex examples follow in Section V, but they are preceded by a discussion of the co-evolutionary principles driving the multidimensional version of the GP algorithm used here (Section IV). We then apply the proposed technique to an engineering design example. We conclude the paper with an assessment of the advantages and drawbacks of the method and outline a number of avenues for further research.

II. Evolution of Intervening Variables Using Genetic Programming

Genetic Programming, an ensemble of techniques originally developed by Koza,[12] applies the Darwinian natural selection paradigm to the evolution of computer programs or symbolic expressions. It is a powerful method for transforming a high level problem statement into a sequence of instructions that can be run on a computer. In striving towards a flexible means of identifying good intervening variables, our interest lies mostly in evolving mathematical expressions – therefore we briefly review this particular aspect of the substantial area of genetic programming.

The standard GP algorithm is population-based. First, an initial pool of candidate solutions is generated, usually randomly. The expressions are then transformed into data structures that are more amenable to computational processing. These strings of operators and operands, referred to as chromosomes, are generated by some encoding procedure – typically Reverse Polish (postfix) notation. New generations of trial expressions are produced using techniques adapted from the Genetic Algorithm literature: selection, crossover, mutation.

As in the case of most evolutionary optimization algorithms, in the GP algorithm the driving force behind the evolution process is an objective function indicating the "fitness" of a candidate solution. In the general template outlined by Koza in his seminal book[13] the evolution of symbolic expressions is usually driven by some type of error measure (whether we are considering symbolic integration, regression, differentiation, etc.). In the case of symbolic regression, the objective function can be a least-squares error function or some other measure that reflects how well the model generalizes.

To illustrate the idea proposed here, consider an approximation model of the form

$$\hat{y}(\mathbf{x}) = \alpha_0 + \sum_{i=1}^{m} \alpha_i \phi_i(\mathbf{x}), \tag{1}$$

where $\mathbf{x}$ is a k-dimensional input vector, $\alpha_i, i = 0, 1, 2, \ldots, m$ are undetermined scalars and $\phi_i(\mathbf{x}), i = 1, 2, \ldots, m$ are known basis functions. A number of surrogate modeling techniques can be written in this form. For example, in the case of linear and quadratic response surface approximation techniques,

*Incidentally, the concept of intervening variables has also been used in the literature to derive two-point approximation techniques, which have been shown to give significantly better accuracy than the conventional first-order Taylor series approximation.[9–11]

the basis functions are polynomials in the input quantities. In the case of radial basis function networks, the bases in Eqn. (1) can be expressed as $\phi_i(||\mathbf{x} - \mathbf{c}||) : \mathbb{R}^k \rightarrow \mathbb{R}$, where $\mathbf{c} \in \mathbb{R}^k$ is the *center* of the basis function. Similarly, feedforward neural networks can also be written in the form of Eqn.(1) by defining appropriate tunable basis functions; see, for example, Ref. 14.

Let us now assume that we apply GP to evolve intervening variables via a transformation $\mathbf{z} = \boldsymbol{\xi}(\mathbf{x})$ to arrive at a modified observational (training) dataset of the form $(\mathbf{z}_i, y_i), i = 1 \ldots n$ (note that only the inputs are transformed). Eqn.(1) can hence be rewritten in terms of the intervening variables as

$$\hat{y}(\mathbf{x}) = \alpha_0 + \sum_{i=1}^{m} \alpha_i \phi_i(\boldsymbol{\xi}(\mathbf{x})). \tag{2}$$

Our goal in using the GP algorithm is to evolve intervening variables in such a way that the model in Eqn.(2) has better generalization properties than the original model in Eqn.(1).

Consider now the case of linear regression, i.e., $m = k$ and $\phi_i(\mathbf{x}) = \mathbf{x}_i$. Here, if the original data $(\mathbf{x}_i, y_i), i = 1 \ldots n$ has a nonlinear input-output relationship, the approximation quality can be rather poor. However, by choosing an appropriate set of intervening variables, it is conceptually possible to arrive at a transformed dataset $(\mathbf{z}_i, y_i), i = 1 \ldots n$ which depicts an approximately linear input-output relationship. Clearly, the least squares error function can be chosen as the objective function which is to be minimized using GP, i.e., we solve the following problem:

$$\underset{\boldsymbol{\xi}(\mathbf{x}), \boldsymbol{\alpha}}{\textbf{Minimize}} : \sum_{j=1}^{n} \left(\hat{y}(\xi(\mathbf{x}_j)) - y_j\right)^2, \tag{3}$$

where $\boldsymbol{\alpha} = \{\alpha_0, \alpha_1, \ldots, \alpha_m\} \in \mathbb{R}^{m+1}$ is the set of undetermined scalars in Eqn.(2)*.

We solve this minimization problem using a two level approach. In the first level, the GP algorithm evolves a population of candidate functions representing $\boldsymbol{\xi}(\mathbf{x})$. In the second level, during fitness evaluation of each candidate solution, the vector $\boldsymbol{\alpha}$ is computed by solving a linear least-squares problem. The resulting value of the training error is the objective function to be minimized by the GP algorithm. Clearly, this approach can eventually lead to a set of intervening variables that approximately linearize the function being modeled. However, in certain cases more care may be required when choosing the objective function if timely convergence is sought. Let us examine some of the issues to be considered here.

Ideally, the objective function minimized by GP should be a good indicator of the generalization error. Since the least-squares error function only minimizes the errors over the training dataset, there exists the possibility that the final model may not generalize well, due to overfitting. It is also worth noting that if $m = n$, irrespective of how the intervening variables are chosen, the least-squares error function is always zero, i.e., the final model is always an interpolator. There are a number of approaches for dealing with such cases and to improve generalization. A possible route is to solve a regularized problem of the form

$$\underset{\boldsymbol{\xi}(\mathbf{x}), \boldsymbol{\alpha}}{\textbf{Minimize}} : \sum_{i=1}^{n} \left(\hat{y}(\xi(\mathbf{x}_i)) - y_i\right)^2 + \lambda g(\boldsymbol{\alpha}), \tag{4}$$

where λ is a regularization parameter and $g(\boldsymbol{\alpha})$ is the regularization function (for example, $g(\boldsymbol{\alpha}) = ||\boldsymbol{\alpha}||$). In the work reported here we use another approach. Whenever the GP solution process of

*A note regarding the implementation of the algorithm: it is often necessary to translate the input space in such a way that the domain does not contain the origin. The purpose of this operation is to reduce the number of candidate solutions that are eliminated prematurely by the selection process due to "division by zero" errors. This, of course, does not change any of the equations presented here – one merely adds a constant to the training data and to the coordinates of points were subsequent predictions are made.

Eqn.(3) converges too slowly, we optimize an alternative objective function, which can be derived using the leave-one-out error measure, an unbiased estimator of the generalization error.[4] Here, the minimization problem to be solved becomes

$$\underset{\boldsymbol{\xi}(\mathbf{x}),\boldsymbol{\alpha}}{\textbf{Minimize}} : \sum_{i=1}^{n} \left(\hat{y}^{-i}(\xi(\mathbf{x}_i)) - y_i\right)^2, \tag{5}$$

where $\hat{y}^{-i}(\xi(\mathbf{x}_i))$ is the predictor obtained by excluding the ith training point. For generalized linear models of the form given in Eqn.(1), this error measure can be approximated efficiently.*

It can be seen clearly from Eqn.(3) that the present approach is different from conventional symbolic function approximation using GP. The latter has been studied in much detail in the literature (see, e.g., Refs. 15, 16); unfortunately, such algorithms often converge relatively slowly for problems with a large number of variables and therefore have found few applications in real-life problems. However, in the present formulation we only use GP to evolve intervening variables, as the top level of the two-level scheme outlined earlier. On the lower level the weights in the approximation model (see Eqn.(2)) are estimated by solving a linear least-squares problem – this can be viewed as a local Lamarckian learning procedure integrated into the GP algorithm.

We now describe a simple, one-variable example to illustrate some of the ideas outlined so far.

III. A Demonstrative Example

The set of training points we have used for our one-variable experiments is shown in Fig. 1, together with the underlying function (a sine wave), which we have sampled at 10 equally spaced points, the first being placed at $x = 0$, while the last of the series is $x = 2\pi$.

We have performed four experiments, generating four different predictors based on this data. The predictors and the corresponding error measures are shown in Fig. 1 and Table 1 respectively. The latter contains four error estimates for each experiment: the Mean Square Error (MSE) of the predictor calculated over the 10 training points and the MSE, average absolute error and L_∞ norm of the predictor calculated over a set of 500 uniformly distributed test points.

First, we have fitted a simple linear regression model using the training data in its original form. As expected, this results in poor generalization (see the first column of Table 1). Slightly better results have been obtained using a simple technique often encountered in the literature: mapping the input space using an intervening variable of the form x^p. This is, essentially, a simple version of the optimization problem set out in Eqn. (3), where a form constraint is imposed on the intervening variable ($\xi(x_i) = x_i^p$) and the MSE is minimized over p. The optimum p in this case is 1.2854 (found by a simple hill-climbing search of the MSE surface) and the predictor over the input space warped using this intervening variable yields the errors listed in the second column of Table 1.

Table 1 Comparison of error measures of the various models fitted to the one-variable example training data.

	Lin. regression	Interv. var. of x^p form	Evolved interv. var.	SR
Test function		$f_1(x) = \sin(x), \quad x \in [0, 2\pi]$		
MSE (training set)	0.2647	0.2587	0.0422	0.2730
MSE (testset)	0.2225	0.2166	0.0301	0.3050
Avg. abs. err. (testset)	0.4216	0.4155	0.1388	0.4768
L_∞ (test set)	0.6859	0.7019	0.3024	0.9159

As this second set of results suggests, for the intervening variable approach to be successful, more flexibility is required in choosing its form. Here we argue that Genetic Programming can be an

*It must be pointed out here that the variance of the leave-one-out error can be high. Therefore, sometimes it may be preferable to use a k-fold cross-validation error measure as the objective function.

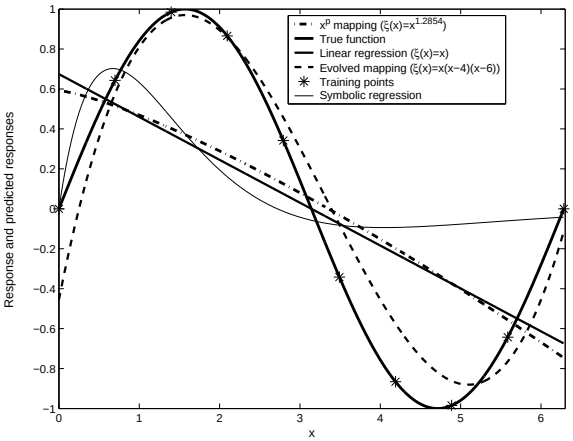

Fig. 1 A demonstrative example: the training points, the underlying (true) response and the responses predicted using four different techniques.

effective, automated means of achieving this. For this example we have run the GP algorithm over 10 generations with a population of 10,000 individuals, the objective function being the MSE of a linear regression model fitted to the training points mapped via the candidate solution being evaluated (i.e., via the proposed intervening variable). In other words, Eqn.(3) is solved again, but, of course, this time the minimization is carried out over a much larger search space. We evolve the symbolic form of $\boldsymbol{\xi}$, allowing it to take its genes from a dictionary of operands and operators comprising: "x", integer constants, "+", "−", "·", "/" and "exp(.)". We note here that in most of the experiments reported in this paper the dictionary also included "sin(.)" and "cos(.)", but since the underlying function here is $\sin(x)$ itself, the inclusion of these building blocks would have made the problem trivial. The best solution obtained after the 10 generations was $z = \xi(x) = x(x-4)(x-6)$. Figure 2 shows the 10 training points again, this time mapped using this intermediate variable. It is clear intuitively that this new set is more amenable to linear regression – column three of Table 1, the list of the errors for the corresponding predictor (also plotted in Figure 1), confirms this.

Let us now consider a final experiment on this data. It can easily be shown that *in a one-dimensional space* the underlying function ($\sin(x)$ in this case) from which the training data is taken, when used as an intervening variable, maps the data into an exactly linear set of responses (i.e., the MSE of the linear regressor vanishes). Therefore, the global optimum found by a standard GP-based symbolic regression (SR) procedure (using an MSE objective function) is also a global optimum of the intervening variable-based technique described above.* However, it is important to point out once more that the two techniques are quite different. They are based on different objective functions and it is also worth noting that there is an infinite number of possible mappings that produce an exactly linear set of responses, most of which are *not* solutions of the classic SR problem (which has a unique exact minimizer). To stress this distinction, we have also applied SR to our initial data, using the same GP algorithm and the same budget of evaluations. The predictor thus found clearly has poorer generalization properties than the one based on intervening variable evolution, as indicated by the last column of Table 1 and Figure 1.

We conclude this brief demonstration with two remarks relating to our implementation. First, it is worth remembering that, given sufficient computing time, it is likely that higher fidelity solutions could be evolved than those obtained here. For example, given the operator set used here, the standard symbolic regression search would probably find an expression similar to the MacLaurin expansion of

*Some authors use the term "Symbolic Regression" to refer to any error-driven search through the space of symbolic expressions (e.g., symbolic integration). Here we use it to mean the search for a symbolic expression to fit observational training data of the $(\mathbf{x}_i, y_i)$ form.

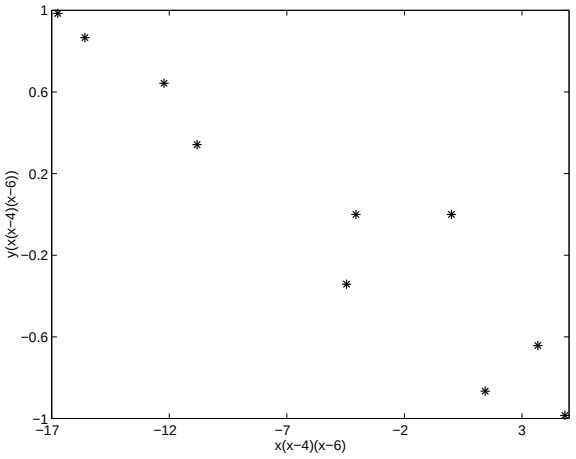

Fig. 2 A demonstrative example: the training points mapped by the intervening variable into the new space.

$\sin(x)$ – see, e.g., Ref. 17. This is a tradeoff that ultimately comes down to the user who has to balance the computational cost of obtaining the training data against the computational cost of evolving the model to approximate it. Secondly, relatively short GP runs, such as those presented here, tend to produce very parsimonious models, which are often linear and the GP algorithm takes a long time to break the dominance of such candidate solutions. In order to discourage such linearity from the very beginning (and thus accelerate the search), we "repair" each candidate solution, by multiplying it by x (i.e., every individual has the "$x\cdot$" genes by default). This is the approach adopted for all experiments presented in this paper.

From the GP point of view, evolving a single intervening variable, as in the example discussed above, poses little technical difficulty. However, multidimensional problems, i.e., the simultaneous evolution of several such expressions, requires an important addition to the standard GP algorithm – we discuss this next.

IV. A Coevolutionary Framework

The techniques of Evolutionary Computation (EC) in general and GP in particular imitate some of the complex processes that create living organisms. However, in doing so, most of these algorithms rarely go beyond copying the fundamental principle of natural selection, namely that complex life results from the non-random survival of randomly varying replicators.[18] One of the simplifications often used by EC practitioners is that the environment, in which the virtual "creatures" evolve, is represented by an entirely *static* figure of merit: the objective function. To be sure, many aspects of the natural environment of a species can be considered to be more or less static (for example, whether its members live in water or on dry land). However, the most important part of the environment in which a creature evolves is the fauna and flora which it shares its habitat with and which it thus *coevolves* with. "Arms races" between predator and prey[19] are a typical example of this, where the evolution of the prey's defences continuously "tracks" that of the predator's weaponry and vice-versa.

Incorporating this idea into some applications of evolutionary algorithms has been one of the most successful nature-inspired EC innovations of recent years. The reader interested in the theoretical aspects and technical details of coevolutionary computation may find the dissertation by Potter[20] a useful reference. As far as applications are concerned, this relatively new paradigm has been used, for example, in multiobjective searches[21] and in MDO type applications.[22] The GP community have also embraced the idea as a natural means of (co)evolving game strategies.[23]

A popular approach in artificial coevolution is to decompose the population into several species[24]

Table 2 Comparison of error measures for the linear regression model fitted to the non-transformed data set and the set expressed in terms of the intervening variable – 2d examples.

	Linear regression	Interv. var. of x^p form	Evolved interv. var.
Test function	$f_2(x_1, x_2) = \sin(x_1) - \cos(x_2), \quad x_{1,2} \in [0, 2\pi]$		
MSE (training set)	0.8052	0.6019	0.0798
MSE (testset)	0.7461	0.7117	0.0692
Avg. abs. err. (testset)	0.7187	0.6954	0.2202
L_∞ (test set)	1.9427	1.6303	0.5724
Test function	$f_3(x_1, x_2) = \frac{1}{f}\left\{-a\exp\left[-b\sqrt{\frac{1}{n}\left(x_1^2 + x_2^2\right)}\right] - \exp\frac{(\cos(cx_1)+\cos(cx_2)}{n} + a + e + d\right\}$, where $a = 20,\ b = 0.2,\ c = 2\pi, d = 5.7,\ e = \exp(1),\ f = 0.8$ and $x_{1,2} \in [-2, 2]$		
MSE (training set)	1.9332	1.6723	0.3249
MSE (testset)	2.0387	1.5974	0.4260
Avg. abs. err. (testset)	1.1810	1.0225	0.5112
L_∞ (test set)	2.5551	2.8002	1.1290
Cross-validation error	106.2949	91.5290	18.1316

and this is the approach we take here. Recollect that in the derivations outlined earlier we wrote the vector of intervening variables in the general form $\mathbf{z} = \boldsymbol{\xi}(\mathbf{x})$. In practice we rewrite the intervening variables as $\mathbf{z} = \{\xi_1(x_1), \xi_2(x_2), \ldots, \xi_k(x_k)\}$, i.e., running the GP algorithm results in a set of one-dimensional functions $\xi_1, \xi_2, \ldots, \xi_k$, which we evolve separately, each dimension having its own population (or species).

The algorithm is, fundamentally, still the classic selecto-recombinative GP heuristic (or, to be precise, k instances of this, running in parallel). The only deviation from the standard scheme is the way in which the objective value of an individual is evaluated. When the objective value of, say, $\xi_i(x_i)$ (residing in population i) is required, we assemble the candidate solution $\mathbf{z}$ by inserting this expression into the i^{th} slot and filling up the remaining $k - i$ slots with representatives of the corresponding populations. The initial (randomly generated) populations are represented in this evaluation process by a randomly chosen individual; in subsequent generations the fittest member of each sub-population is used.

We now continue with an experimental test of the performance of this co-evolutionary scheme – first, using analytical test functions.

V. Two Analytical Test Functions

As a first application of the coevolution-based search for intermediate variables, let us consider the simple analytical test function shown in Fig. 3a. We have generated a 16 point training set, using a Morris-optimal[25] Latin Hypercube experimental design. The 1600 test points used for the off-line assessment of the predictors based on this training data were arranged in a full factorial design. As far as the GP setup is concerned, here and in subsequent examples the search was performed over 10 generations with 2500 individuals in each population.

The top section of Table 2 compares the generalization properties of the models generated with three approaches applied to this data. The first column contains the error measures of linear regression applied to the original data. The second technique investigated here is linear regression using a pair of intervening variables of the form $\mathbf{z} = \{\xi_1(x_1), \xi_2(x_2)\} = \{x_1^{p_1}, x_2^{p_2}\}$. Solving Eqn.(3) over p_1 and p_2 using a conjugate gradient search yields $p_1 = 1.0542$ and $p_2 = 0.1110$ and an MSE over the training data of 0.6019. The generalization measures are listed in column two of the same table.

Finally, the results of applying the GP-based intervening variable technique point again at the advantages of flexibility in building such a mapping: the MSE, as measured over the test data, is an order of magnitude lower than in the previous case. The intervening variables evolved by the GP were $\mathbf{z} = \{\xi_1(x_1), \xi_2(x_2)\} = \{x_1^3 - 9x_1^2 + 17x_1, -3x_2^2 + 19x_2\}$. As in the case of the demonstrative

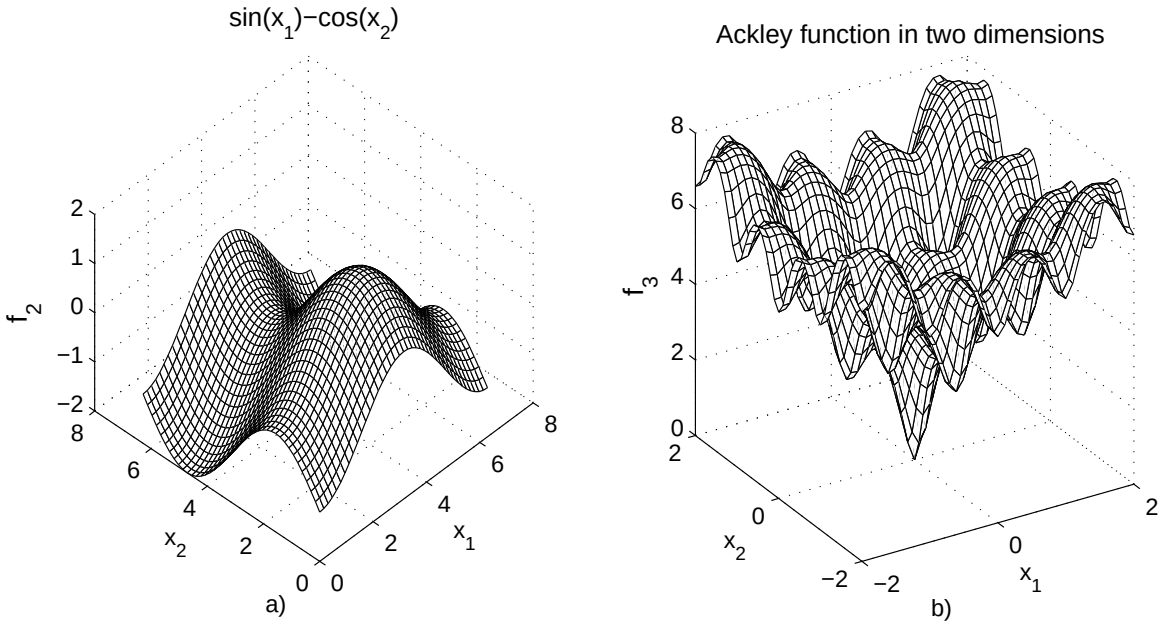

Fig. 3 The two-variable test functions.

example of Section III, "sin(.)" and "cos(.)" were excluded from the operator dictionary available to the GP algorithm (a run with a dictionary including "sin(.)" and "cos(.)" found the exact solution $\mathbf{z} = \{\xi_1(x_1), \xi_2(x_2)\} = \{\sin(x_1), \cos(x_2)\}$ in the second generation).

The next analytical test case, the two-variable Ackley function depicted in Fig. 3b, proved to be much more challenging. Keeping the 40^2 full factorial design for the generation of the test data, this time we adopted a larger training set, a 49-point Latin Hypercube. We carried out the same three experiments (results shown in the bottom section of Table 2). On this occasion, the convergence of the GP method using the MSE-based objective was slow so we opted for the solution of Eqn.(5) instead, i.e., for the minimization of the leave-one-out cross-validation error (hence, this criterion is also included in the results table).

For the $\mathbf{z} = \{\xi_1(x_1), \xi_2(x_2)\} = \{x_1^{p_1}, x_2^{p_2}\}$ intervening variable approach we have found that $p_1 = p_2 = 6.0313$ minimizes the cross-validation error over the training data and the generalization error measures are shown in the second column of Table 2.

The GP-based process generated the intervening variables $\mathbf{z} = \{\xi_1(x_1), \xi_2(x_2)\} = \{x_1(\sin(x_1^2 + 4x_1) - x_1 + 2), x_2 \cos(x_2 - 3\sin x_2)\}$ and, as previously, these improved the generalization properties of the linear model, although not quite as substantially as with the previous test function. Also, as the relatively high cross-validation error suggests, the brief GP run was not sufficient for the mapping-based model to capture all modes of the Ackley function.

The attentive reader will have noticed that so far we have only dealt with separable test functions, for which there exists a vector of intervening variables that will map the original data into a space where the MSE of the linear regressor will be zero (although it is unlikely that the GP algorithm will find this exact expression in a reasonable amount of time for all but the most trivial cases). To complete our empirical investigation, we now look at a non-separable example.

VI. A "Real Life" Design Problem

The pair of testcases we discuss here is based on the optimization of the frequency response of a two-dimensional structure of the sort that may be found in girder bridges, tower cranes, satellite

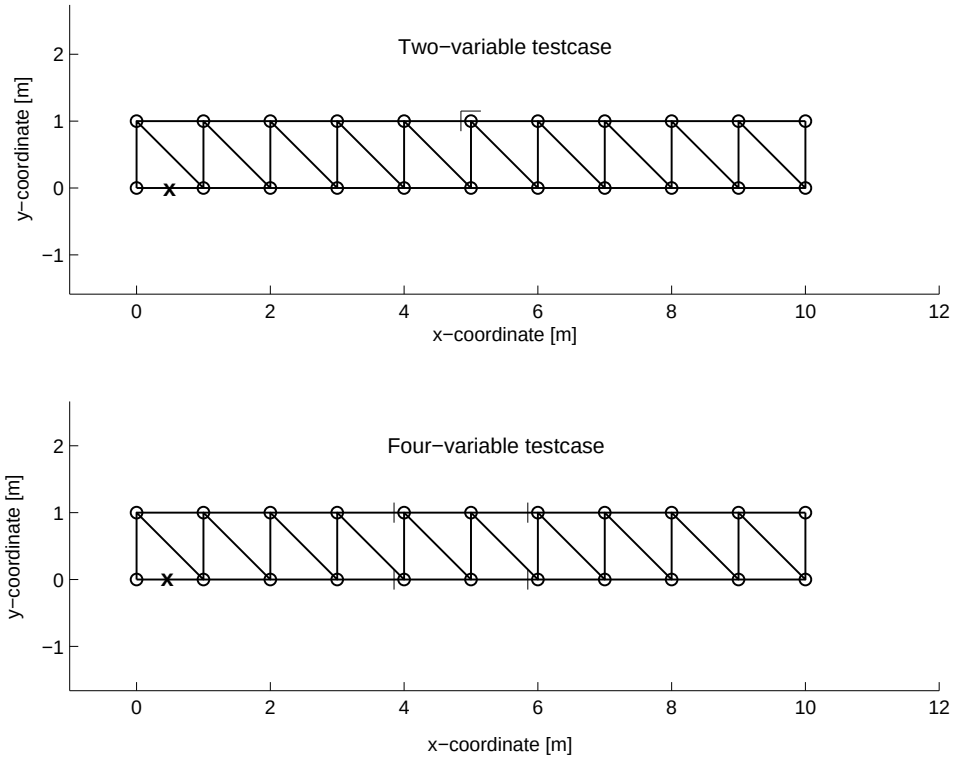

Fig. 4 The two structural optimization testcases. The horizontal and vertical lines around some of the joints indicate which joints and to what extent are allowed to move during the design process. The excitation point is marked by an 'x' on each beam.

booms, etc.[26] It consists of 40 individual Euler-Bernoulli beams connected at 20 joints. Each of the 40 beams has the same properties per unit length.

Initially the truss was designed and analyzed for a regular geometry, where each beam was either 1 m or 1.414 m in length (see Fig. 4). The joints at points (0,0) and (0,1) are fixed, i.e., they are fully restrained in all degrees of freedom. All other joints are free to move. The structure is excited by a point transverse force applied halfway between points (0,0) and (1,0) (as indicated by the 'x' on the figures). The vibrational energy level was calculated for the right-hand end vertical beam[27] * – the minimization of this frequency averaged response in the range 150-250 Hz was chosen as the objective.

We have constructed two different testcases: a two-variable one (denoted by f_4), where where the x and y coordinates of the top mid-span joint were allowed to vary by ± 0.15m (Fig. 4a) and a four-variable one (f_5), where the y coordinates of the joints either side of the beam connecting the two mid-span joints were allowed the same range of movement (Fig. 4b). The objectives are denoted by f_4 and f_5, respectively.

The experiments and the structure of the results table (Table 3) are the same as before. The sizes of the training sets and test sets were 16 and 1600 respectively for the two-variable example and 120 and 5000 for the four-dimensional test case. The optimization of Eqn. (5) over the exponent values of the x^p intervening variable yielded $p_1 = 4.5121$ and $p_2 = 0.6897$ and $p_1 = 0.2670$, $p_2 = 2.3351$, $p_3 = 0.1$ and $p_4 = 0.1095$ respectively. Allowing more flexibility in the choice of the intervening variables, i.e., applying the GP approach, led to the "discovery" of the following mappings: $\mathbf{z} = \{\xi_1(x_1), \xi_2(x_2)\} = \{x_1 \exp(x_1 - \exp(3x_1))\,,\, 5x_2^5 - 2x_2^3 + x_2\}$ for the two-variable case and $\mathbf{z} = \{\xi_1(x_1), \ldots \xi_4(x_4)\} = \{1/x_1\,,\, 1\,,\, 2x_3^2 - \frac{5}{2}x_3\,,\, x_4^2 - \frac{5}{4}x_4\}$ for the four-dimensional problem. Once again, as the comparison

*We note here that the results of the analysis have been validated experimentally.[28]

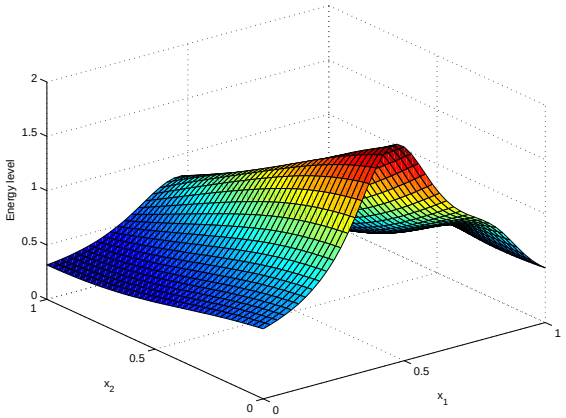

Fig. 5 The objective function landscape of the two-dimensional truss example (f_4).

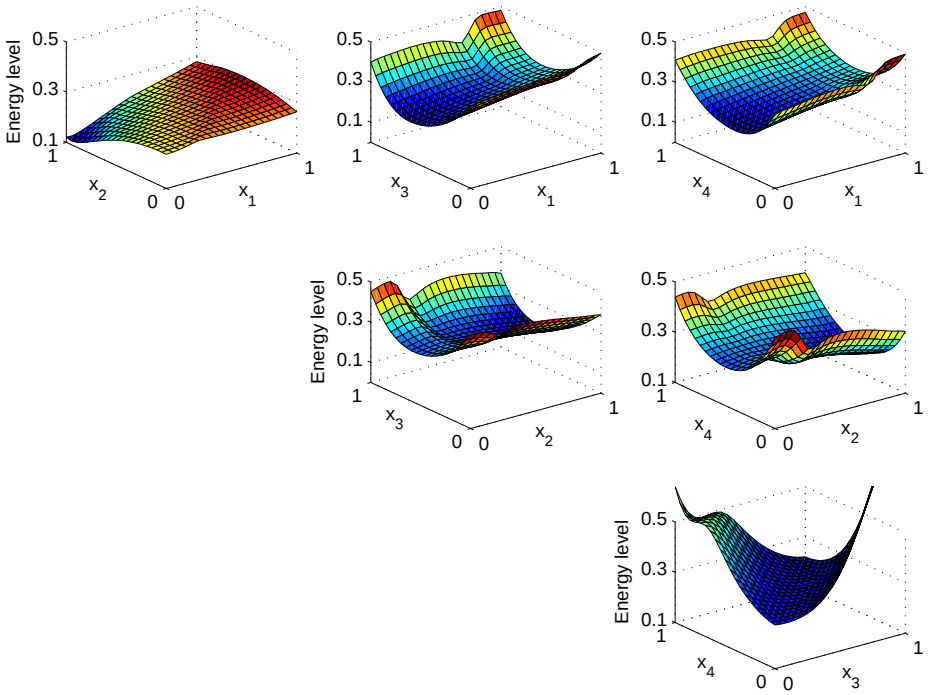

Fig. 6 Two-variable sections through the objective function landscape of the four-dimensional truss example (f_5). Each section was computed by sweeping the range $[0, 1]$ with two of the variables and holding the remaining two at 0.5 (i.e., at the values corresponding to the baseline truss shown in Fig. 4).

Table 3 Comparison of error measures for the linear regression model fitted to the non-transformed data set and the set expressed in terms of the intervening variable – truss beam example.

	Linear regression	Interv. var. of x^p form	Evolved interv. var.
Test function		$f_4(x_1, x_2)$, $x_{1,2} \in [0, 1]$	
MSE (training set)	0.0949	0.0741	0.0170
MSE (test set)	0.0998	0.0770	0.0271
Avg. abs. err. (test set)	0.2635	0.2176	0.1343
L_∞ (test set)	0.8354	0.7109	0.5258
Cross-validation error	2.3202	1.6744	0.4430
Test function		$f_5(x_1, \ldots x_4)$, $x_{1\ldots4} \in [0, 1]$	
MSE (training set)	0.0189	0.0174	0.0131
MSE (test set)	0.0276	0.0381	0.0239
Avg. abs. err. (test set)	0.1275	0.1426	0.1185
L_∞ (test set)	0.2156	0.7866	0.4553
Cross-validation error	2.4951	2.3493	1.7841

criteria listed in Table 3 demonstrate, the GP-evolved intervening variables improve the generalization properties of the linear predictor, although the change is less marked in the case of the four-dimensional example.

VII. Conclusions and Future Directions

The selection of intervening variables for response surface approximation is usually based on problem-specific knowledge (as is often the case in aerodynamic optimization). Where no such knowledge is available, designers often resort to using a parametric intervening variable (usually of a simple form, such as x^p), which is optimized over its parameter space. In this paper we have proposed the use of Genetic Programming as a more flexible means of selecting intervening variables and experimented with the idea on a set of test cases.

Although some of the results reported here are promising, we cannot claim at this stage that they are of universal value and, consequently, we shall not draw any far-reaching conclusions based on them. They merely serve as "proof-of-concept" experiments, which later work can build on. In our view, the next step would be a more thorough empirical study, using a wider range of test functions and looking into issues such as the variance of the quality of the results (arising from the stochastic nature of the GP paradigm), the choice of the operator-dictionary that the GP algorithm can draw on, etc. Also, more sophisticated GP algorithms could be considered, perhaps using specialized operators – our experiments were conducted using a standard, general purpose GP code. We have briefly touched on the issue of balancing intervening variable evolution time against the time required to obtain an extra training point. This is another aspect (arising, of course, with almost any global modeling technique) that a more detailed study should look into.

Although our central aim was to facilitate the construction of accurate linear predictors, the more detailed analyses suggested above could be of use for the wider community as well, as the approach outlined here appears to offer significant time savings over standard symbolic regression curve fitting.

As far as extensions of the method are concerned, an immediate application could be its use to enhance the generalization properties of *nonlinear* models. For example, the same framework could be used to construct better DACE models, by replacing the current local (Lamarckian) learning component of the GP search with a maximization in hyperparameter space of the sample likelihood of the model. In a machine learning context, the technique could be used to automate *feature extraction* – again, the general framework could be the same as here, but the operand-dictionary of the GP algorithm would include all k variables for each expression.

References

[1]Montgomery, D. C. and Myers, R. H., *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*, Wiley-Interscience, 1995.

[2]Bishop, C., *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[3]Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P., "Design and Analysis of Computer Experiments (with discussion)," *Statistical Science*, Vol. 4, 1989, pp. 409–435.

[4]Vapnik, V., *Statistical Learning Theory*, John-Wiley and Sons, New York, 1998.

[5]Booker, A. J., Dennis, J. E., Frank, P. D., Serafini, D. B., Torczon, V., and Trosset, M. W., "A Rigorous Framework for Optimization of Expensive Functions by Surrogates Structural Optimization," *Structural Optimization*, Vol. 17, No. 1, 1998, pp. 1–13.

[6]Nair, P. B., Choudhury, C., and Keane, A., "A Bayesian Framework for Uncertainty Analysis Using Deterministic Black-Box Simulation Codes," *AIAA Paper 2001-1676*, 2001.

[7]Ong, Y., Nair, P. B., and Keane, A., "Evolutionary Optimization of Computationally Expensive Problems via Surrogate Modeling," *AIAA Journal*, Vol. 40, No. 4, 2003, pp. 687–696.

[8]Schmidt, L. A. and Miura, H., "Approximation Concepts For Efficient Structural Synthesis," Report CR-2552, NASA, 1976.

[9]Fadel, G. M., Riley, M. F., and Barthelemy, J. F. M., "Two-Point Exponential Approximation Method for Structural Optimization," *Structural Optimization*, Vol. 2, No. 2, 1990, pp. 117–124.

[10]Wang, L. P. and Grandhi, R. V., "Improved Two-Point Function Approximation for Design Optimization," *AIAA Journal*, Vol. 33, No. 9, 1995, pp. 1720–1727.

[11]S., X. and Grandhi, R. V., "Effective Two-Point Function Approximation for Design Optimization," *AIAA Journal*, Vol. 36, No. 12, 1998, pp. 2269–2275.

[12]Koza, J. R., "Genetically Breeding Populations of Computer Programs to Solve Problems in Artificial Intelligence," *In Proceedings of the Second International Conference on Tools for AI, Herndon, Virginia*, 1990, pp. 819–827.

[13]Koza, J. R., *Genetic Programming – On the Programming of Computers by Means of Natural Selection*, Vol. I., The MIT Press, 1992.

[14]Nair, P. B., "Physics-Based Surrogate Modeling of Parameterized PDEs for Optimization and Uncertainty Analysis," *AIAA Paper 2002-1586*, 2002.

[15]Salhi, A., Glaser, H., and Roure, D. D., "Parallel Implementation of a Genetic-programming Based Tool for Symbolic Regression," *Information Processing Letters*, Vol. 66, 1998, pp. 299–307.

[16]Nikolaev, N. and Iba, H., "Regularization Approach to Inductive Genetic Programming," *IEEE Transaction on Evolutionary Computation*, Vol. 5, No. 4, 2001, pp. pp.359–375.

[17]Bearpark, K., *Learning and Memory in Genetic Programming*, Phd thesis, University of Southampton, Southampton, UK, 2000.

[18]Dawkins, R., *The Blind Watchmaker*, W.W. Norton & Company, 1996.

[19]Dawkins, R. and Krebs, J. R., "Arms Races Between and Within Species," *Proceedings of the Royal Society of London B*, Vol. 205, 1979, pp. 489–511.

[20]Potter, M. A., *The Design and Analysis of a Computational Model of Cooperative Evolution*, PhD thesis, George Mason University, Fairfax, VA, 1997.

[21]Husbands, P., "Distributed Coevolutionary Genetic Algorithms for Multi-criteria and Multi-constraint Optimization," *Evolutionary Computing, Lecture Notes in Computer Science, Springer-Verlag*, Vol. 865, 1994, pp. 150–165.

[22]Nair, P. B. and Keane, A. J., "Coevolutionary Architecture for Distributed Optimization of Complex Coupled Systems," *AIAA Journal*, Vol. 40, No. 7, 2002, pp. 1434–1443.

[23]Koza, J. R., "Genetic Evolution and Co-evolution of Computer Programs," *Artificial Life II*, edited by C. T. C. Langton, J. D. Farmer, and S. Rasmussen, Vol. X, Addison-Wesley, Santa Fe Institute, New Mexico, USA, 1990 1991, pp. 603–629.

[24]Potter, M. A. and DeJong, K. A., "A Cooperative Coevolutionary Approach to Function Optimization," *Proceedings of the Third Conference on Parallell Problem Solving From Nature, Springer-Verlag*, 1994, pp. 249–257.

[25]Morris, M. D. and Mitchell, T. J., "Exploratory Designs for Computer Experiments," *Journal of Statistical Planning and Inference*, Vol. 43, 1995, pp. 381–402.

[26]Renton, J. D., *Elastic Beams and Frames*, Camford Books, 1999.

[27]Keane, A. J., "Passive Vibration Control via Unusual Geometries: the Application of Genetic Algorithm Optimization to Structural Design," *Journal of Sound and Vibrations*, Vol. 185, No. 3, 1995, pp. 441–453.

[28]Keane, A. J. and Bright, A. P., "Passive Vibration Control via Unusual Geometries: Experiments on Model Aerospace Structures," *Journal of Sound and Vibrations*, Vol. 190, No. 4, 1996, pp. 713–719.