Surrogate Based EA for Expensive Optimization Problems

Maumita Bhattacharya, *Member, IEEE*

***Abstract*—Real life optimization problems often require finding optimal solution to complex high dimensional, multimodal problems involving computationally very expensive fitness function evaluations. Use of a population based iterative technique like evolutionary algorithms in such problem domains is thus practically prohibitive. An attractive alternative is to build surrogates or use an approximation of the actual fitness functions to be evaluated. Many regression and interpolation tools are available to build such meta models. This paper briefly discusses the architectures and use of such meta-modeling tools in an evolutionary optimization context. We further present an evolutionary algorithm framework which involves use of surrogate models for fitness function evaluation. The original framework namely, the Dynamic Approximate Fitness based Hybrid EA (DAFHEA) model [14] reduces computation time by controlled use of meta-models (in this case approximate model generated by Support Vector Machine regression) to partially replace the actual function evaluation by approximate function evaluation. However, the underlying assumption in DAFHEA is that the training samples for the meta-model are generated from a single uniform model. This does not take into account uncertain scenarios involving noisy fitness functions. The enhanced model, DAFHEA-II, incorporates a multiple-model based learning approach for the support vector machine approximator to counter effects of noise [15]. Empirical results obtained by evaluating the frameworks using several benchmark functions (both non-noisy and noisy versions) are presented.**

I. Background

THE optimization of complex, high dimensional, multimodal problems often poses a problem which in fact depends on the structure inherent in the problems. One of the most critical of the features is that the functions used to define such optimization problems are often computationally intensive. In such problems, the run-time for a single function evaluation could be in the range from a fraction of a second to hours of supercomputer time. Considering such prohibitive computational costs, a feasible alternative is to build approximate models, within an optimization context, since these approximate models are order of magnitude cheaper to run compared to the actual function evaluations [8,10,11]. Many regression and interpolation tools could be used to construct such meta models, (e.g. least square regression, back propagating artificial neural network, response surface models, etc.) which provide *less accurate*, but *more efficient* (in terms of computational cost) measures of the *merit* of the fitness functions. However, accuracy of the result is a major risk involved in using meta-models to replace actual function evaluation [23, 25, 26, and 27]. When it is infeasible to precisely judge when, where and how much of such replacement is optimal, using a controlled approach holds the answer.

Use of approximate model to speed up optimization dates all the way back to the sixties [14]. The most widely used models being Response Surface Methodology [17], Krieging models [9] and artificial neural network models [5]. The concepts of using approximate model vary in levels of approximation (*Problem approximation, Functional approximation, and Evolutionary approximation*), model incorporation mechanism and model management techniques [27]. In the multidisciplinary optimization (MDO) community, primarily response surface analysis and polynomial fitting techniques are used to build the approximate models [16, 23]. They are not well suited for high dimensional multimodal problems as they generally carry out approximation using simple quadratic models. In another approach, multilevel search strategies are developed using special relationship between the approximate and the actual model. An interesting class of such models focuses on having many islands using low accuracy/cheap evaluation models with small number of finite elements that progressively propagate individuals to fewer islands using more accurate/expensive evaluations [7]. As is observed in [27], this approach may suffer from lower complexity/cheap islands having false optima whose fitness values are higher than those in the higher complexity/expensive islands. Rasheed et al. in [10, 11], uses a method of maintaining a large sample of points divided into clusters. Least square quadratic approximations are periodically formed of the entire sample as well as the big clusters. Problem of unevaluable points was taken into account as a design aspect. However, it is only logical to accept that true evaluation should be used along with approximation for reliable results in most practical situations. Another approach using population clustering is that of *fitness imitation* [27]. Here, the population is clustered into several groups and true evaluation is done only for the cluster representative [8]. The fitness value of other members of the same cluster is estimated by a distance measure. The method may be too simplistic to be reliable, where the population landscape is a complex, multimodal one. Jin et al. in [25, 26] analysed the convergence property of approximate fitness based evolutionary algorithm. It has been observed that incorrect convergence can occur due to false optima

This work was supported in part by a research grant of Charles Sturt University, Australia.

Maumita Bhattacharya is with the School of Business and Information Technology, Charles Sturt University, NSW 2640 AUSTRALIA (corresponding author to provide phone: 61 2 60519619; e-mail: maumita.bhattacharya@ ieee.org).

introduced by the approximate model. Two *controlled evolution* strategies have been introduced. In this approach, new solutions (offspring) can be (pre)-evaluated by the model. The (pre)-evaluation can be used to indicate promising solutions. It is not clear however, how to decide on the optimal fraction of the new individuals for which true evaluation should be done [6]. In an alternative approach, the optimum is first searched on the model. The obtained optimum is then evaluated on the objective function and added to the training data of the model [1, 22, and 6]. Yet in another approach as proposed in [25], a regularization technique is used to eliminate false minima.

It is obvious that incorporation of approximate models may be one of the most promising approaches to realistically use EA to solve complex real life optimization problems, especially where: (i). Fitness computation is highly time-consuming, (ii). Explicit model for fitness computation is absent, (iii). Environment of the evolutionary algorithm is noisy etc. However, considering the obvious risk involved in such approach, an EA with efficient control strategy for the approximate model and robust performance is welcome.

The proposed hybrid evolutionary algorithm framework, DAFHEA (dynamic approximate fitness based hybrid evolutionary algorithm) replaces expensive function evaluation by its support vector machine (SVM) approximation. The concept of *merit function* [22] is borrowed to maintain diversity in the solution space using approximate knowledge. However, the assumption used in the original DAFHEA is that the training samples for the meta-model are generated from a single uniform model. This does not cover situations, where information from variable input dimensions and noisy data is involved. DAFHEA-II attempts to correct this by using a multi-model regression approach. The multiple models are estimated by successive application of the SVM regression algorithm. Retraining of the model is done in a periodic fashion.

The original DAFHEA framework [14] is similar to other existing models in that it uses an approximation model to partially replace expensive fitness evaluation in evolutionary algorithm. An *explicit* control strategy (*a cluster-based on-line learning technique*) to improve reliability of using such approximate models to reduce expensive function evaluations was introduced. Also the approximate knowledge thus generated is exploited to avoid premature convergence (one of the major impediments of using evolutionary algorithm to solve complex real life optimization problems). However, the major constraint associated with DAFHEA is that it treats the solution space as one comprising of information coming from a uniform model. Situations like model formation involving variable input dimensions and noisy data certainly can not be covered by this assumption. DAFHEA-II addresses this issue by using a multiple model regression approach for the SVM approximator.

The rest of the paper is arranged as follows: Section II presents a brief description of popular surrogate generation tools. The basic concepts and the functional details of DAFHEA and DAFHEA-II (the enhanced version of DAFHEA) are presented in Section III, IV and Section V. Experiment results and discussions are presented in Section VI. Finally we conclude in Section VII.

II. META MODEL GENERATION TOOLS

A wide variety of empirical tools are available to generate functional approximation models [17, 9, 5]. Some of the popular approximations modeling tools are described below:

A. Polynomial Approximation Models

One of the popular and simple meta modeling methods is the *linear polynomial regression* model. This method involves finding the coefficients of the linearly combined terms in a given low order polynomial or a vector of polynomials.

The second-order polynomial approximation model has the following form:

$$\hat{y} = c_0 + \sum_{1 \le i \le n} c x_i + \sum_{1 \le i \le j \le n} c_{(n-1+i+j)} x_i x_j \quad (1)$$

where, c_0, c_i and $c_{(n-1+i+j)}$ are the coefficients to be estimated. Note that, there are $n_t = (n+1)(n+2)/2$ modeling terms in the quadratic polynomial, where, n is the number of input variables.

Both least square method (LSM) and the gradient method can be used to estimate the unknown coefficients.

1) Least Square Method

If n_s is the number of samples drawn from the original function and $p = 1, \ldots, n_s$ then the polynomial model in matrix notation has the following form:

$$y^{(p)} = \mathbf{c}^{\mathbf{T}} \bar{\mathbf{x}}^{(\mathbf{P})} \quad (2)$$

where, $\mathbf{c}$ is the vector of n_t unknown coefficients to be estimated.

$$\mathbf{c} = \left\lfloor c_0, c_1, \ldots, c_{n_t - 1} \right\rfloor \quad (3)$$

and $\bar{\mathbf{x}}^{(\mathbf{P})}$, the polynomial model, is a vector of length n_t.

Estimating the unknown coefficients will require that $n_s \ge n_t$. The estimation problem can now be formulated as below:

$$\mathbf{y} \approx \mathbf{X}\hat{\mathbf{c}} \quad (4)$$

where, $\mathbf{y}$ is the vector of n_s response values,

$$\mathbf{y} = \left[y^{(1)}, y^{(2)}, \ldots, y^{(n_s)} \right]$$

and $\mathbf{X}$ is the matrix of p row vectors with assumed rank of n_t,

$$\mathbf{X}=\begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & \left(x_{n_v}^{(1)}\right)^2 \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & \left(x_{n_v}^{(2)}\right)^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n_s)} & x_2^{(n_s)} & \cdots & \left(x_{n_v}^{(n_s)}\right)^2 \end{bmatrix}$$

The unique LSM solution is as follows,

$$\hat{\mathbf{c}}=\left(\mathbf{X}^{\mathbf{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{y} \tag{5}$$

where, $\hat{\mathbf{c}}$ denotes the estimate of $\mathbf{c}$ and $\left(\mathbf{X}^{\mathbf{T}}\mathbf{X}\right)^{-1}$ exists if the rows of $\mathbf{X}$ are linearly independent.

2) Gradient Method

The least square method is unsuitable for high dimensional problems due to high computational expenses. Gradient method is a suitable alternative to address this problem [27].

B. The DACE Model

This is also known as Gaussian process regression in the neural network literature and Kriging in the geostatistics literature. The DACE model may be understood as a two-component model that includes a global model and a localized "deviation". It can be expressed as,

$$y(x)=f(x)+Z(x) \tag{6}$$

where, $f(x)$ represents a global model of the original function and $Z(x)$ is a Gaussian random function with zero mean and non-zero covariance that creates a localized deviation from the global model. $f(x)$ is usually a polynomial and can be considered to be an underlying constant β. Then the representation of DACE model becomes,

$$y(x)=\beta+Z(x) \tag{7}$$

The covariance matrix of $Z(x)$ is expressed as,

$$Cov\left[Z\left(x^{(i)}\right),Z\left(x^{(j)}\right)\right]=\sigma^2\mathbf{R}\left[R\left(x^{(i)},x^{(j)}\right)\right] \tag{8}$$
$$j,k=1,\cdots,n_s$$

where, $\mathbf{R}$ is the symmetric correlation matrix with values of unity along the diagonal and dimension of $n_s\times n_s$. R is the correlation function between any two of n_s samples. The user may select the form of the correlation function R. A choice for R often found in statistical literature is an exponential function,

$$R\left(x^{(i)},x^{(j)}\right)=\exp\left[-\sum_{k=1}^{n}\theta_k\left|x_k^{(i)}-x_k^{(j)}\right|^2\right] \tag{9}$$

where, θ_k is the vector of the unknown correlation parameters. $x_k^{(i)}$ and $x_k^{(j)}$ are the k^{th} component of the sample point $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$. The prediction of $\mathrm{y}(\mathbf{x})$ can thus be represented as,

$$\hat{y}(x)=\hat{\beta}+\mathbf{r}^T(\mathbf{x})\mathbf{R}^{-1}\left(\mathbf{y}-\hat{\beta}\mathbf{f}\right) \tag{10}$$

where, $\hat{\beta}$ is unknown and both $\mathbf{r}(\mathbf{x})$ and $\mathbf{R}$ depend on the unknown parameter θ. The vector $\mathbf{f}$ has length n_s, with all entries equal to unity. $\mathbf{y}$ is a vector of length n_s and $\hat{y}$ is the estimated value of y given the n_s samples and the current input $\mathbf{x}$. $\mathbf{r}$ is the correlation vector of length n_s between the given input $\mathbf{x}$ and the samples $\left\{\mathbf{x}^{(1)},\cdots,\mathbf{x}^{(n_s)}\right\}$

$$\mathbf{r}^T(\mathbf{x})=\left[R\left(\mathbf{x},\mathbf{x}^{(1)}\right),R\left(\mathbf{x},\mathbf{x}^{(2)}\right),\cdots,R\left(\mathbf{x},\mathbf{x}^{(n_s)}\right)\right]^T \tag{11}$$

The estimation of the parameters can be carried out using the maximum likelihood method.

C. Artificial Neural Networks

The artificial neural networks consist of a large number of highly interconnected processing units, each aggregating information from a large number of connected peers.

The feed forward multilayer perceptrons and the radial basis function networks are the popular ones.

1) Multilayer Perceptrons

A multilayer perceptron with one input layer, two hidden layers and one output neuron can be expressed as [27],

$$y=\sum_{t=1}^{L}v_t f\left(\sum_{h=1}^{K}w_{ht}^{(2)}f\left(\sum_{i=1}^{n}w_{ih}^{(1)}x_i\right)\right) \tag{12}$$

where, n is the number of inputs, K and L are the number of hidden nodes, and $f(\cdot)$ is called the activation function, which usually is the logistic function.

$$f(z)=\frac{1}{1+\mathbf{e}^{-\mathbf{az}}} \tag{13}$$

where, $\mathbf{a}$ is a constant.

2) Radial Basis Function Network

The radial basis functions are powerful interpolation tools for multidimensional problems. The output of a radial basis function network is a weighted sum of radial basis functions $\phi(\cdot)$, each being a local function (usually Gaussian) of the distance $\left(\left\|\mathbf{x}-\mathbf{c}_j\right\|\right)$ between the input vector $\mathbf{x}$ and a "basis function center", $\mathbf{c}_j$,

$j=1,\cdots,n_s$ and n_s is the number of radial basis functions. The un-normalized RBF network is of the form,

$$y(\mathbf{x}) = \sum_{j=1}^{n_s} w_j \phi\left(\left\|\mathbf{x} - \mathbf{c}_j\right\|\right) \quad (14)$$

where, $\|\cdot\|$ is typically an Euclidean norm and w_j are the weights or the unknown coefficients to be determined.

The normalized RBF network can be expressed as,

$$y(\mathbf{x}) = \frac{\sum_{j=1}^{n_s} w_j \phi\left(\left\|\mathbf{x} - \mathbf{c}_j\right\|\right)}{\sum_{j=1}^{n_s} \phi\left(\left\|\mathbf{x} - \mathbf{c}_j\right\|\right)} = \sum_{j=1}^{n_s} w_j u\left(\left\|\mathbf{x} - \mathbf{c}_j\right\|\right) \quad (15)$$

where, $u\left(\left\|\mathbf{x} - \mathbf{c}_j\right\|\right)$ is known as the normalized radial basis function.

D. Support Vector Machines

The conceptual background of single approximation model building using support vector machines has been outlined in Section V. A.

The subsequent sections describe an EA framework that use SVM regression to generate the surrogates.

III. The DAFHEA Framework

The DAFHEA framework includes a global model of genetic algorithm (GA), hybridized with support vector machine (SVM) as the approximation tool. Expensive fitness evaluation of individuals as required in traditional evolutionary algorithm is partially replaced by a SVM approximation (regression) model. Explicit control strategies are used for *evolution control,* leading to considerable speedup without compromising heavily on solution accuracy. Also the approximate knowledge about the solution space generated is used to maintain population diversity to avoid premature convergence.

While approximation is not a new idea in accelerating iterative optimization process, DAFHEA focuses on controlled speedup to avoid detrimental effects of approximation and also exploiting approximate knowledge to improve optimization solution. The detailed algorithm structure of DAFHEA is described in our earlier work in [14].

IV. The DAFHEA-II Framework

As in the original DAFHEA framework, DAFHEA-II [Figure 1] includes a global model of genetic algorithm (GA), hybridised with support vector machine (SVM) as the approximation tool. Expensive fitness evaluation of individuals as required in traditional evolutionary algorithm is partially replaced by SVM approximation (regression) models. *Evolution control* is implemented by periodic expensive evaluations, leading to considerable speedup without compromising heavily on solution accuracy. Also the approximate knowledge about the solution space generated is used to maintain population diversity to avoid premature convergence.

DAFHEA-II is specifically suited for applications involving information that could be considered generated by more than one model. As in original DAFHEA, this framework also focuses on controlled speedup to avoid detrimental effects of approximation and exploiting approximate knowledge to improve optimization solution. The following section presents the basic algorithm structure of DAFHEA-II, while Section IV explains its major functional aspects.

A. Basic Algorithm Structure

The proposed DAFHEA-II framework is introduced in the context of unconstrained optimization problems. Figure (1) schematically presents the algorithm.

```
/* The basic algorithm for the DAFHEA-II
framework */
Procedure DAFHEA_II
{
  {
  initialize population matrix, gen = 0
  and  set α, β
  call actual function evaluation
  call Procedure train_SVM to generate
  approximation models
  while ( gen < α )
    {
      gen = gen + 1
      rank solutions based on fitness
      retain actual elite
      apply crossover and mutation to
      generate offspring
      call Procedure predict_SVM to
      approximate the fitness of the
      offspring
      retain approximate elite
      if ( gen mod β = 0 ) then
        {
        call actual function evaluation
        call Procedure train_SVM
        }
      }
      rank  solutions based on fitness
      get the best solution
    }
/* This procedure estimates multiple
models from the training data set */
Procedure train_SVM
{
  initialize data set=population in
  current generation with fitness
  resulting from actual evaluation
  while (!stopping criterion)
  {
/* Estimate the dominant model
describing majority of the candidates in
data set */
```

```
      apply robust regression to data
      set
/* Partition the available data */
      {
      analyse available data to separate
      others from majority based on
      their
      distance from the dominant model
      remove subset of data generated by
      the dominant model from the data
      set
      }
  }
}
/* This procedure selects the most
likely model for each member of the
population to find its corresponding
estimate */

Procedure predict_SVM
{
  while (population member)
  {
  determine appropriate model for the
  given test sample z = (x, y) using a
  distance measure from each model
  }
}
}
```

Fig. 1 The DAFHEA-II Framework

In the above framework (see Figure 1) α is the number of predetermined generations and β is the predetermined retraining frequency.

B. The Implementation of the DAFHEA-II Framework

The detailed implementation of the DAFHEA-II [15] framework is as given below:

Step One: Create a random population of N_c individuals, where, $N_c = 5 * N_a$ and $N_a =$ actual initial population size.

Step Two: Evaluate N_c individual using actual expensive function evaluation. Build the SVM approximate models using the candidate solutions as input and the actual fitness (expensive function evaluation values) as targets forming the training set for *off-line training*. Details of the Multiple Model Formation technique is described in Section IV.

Step Three: Select N_a best individual out of N_c evaluated individuals to form the initial GA population.

Remarks: The idea behind using five times the actual EA population size (as explained in *Step One*) is to make the approximation model sufficiently representative at least initially. Since initial EA population is formed with N_a best individuals out of these N_c individuals, with high recombination and low mutation rates, the EA population in first few generations is unlikely to drift much from its initial locality. Thus it is expected that large number of samples used in building the approximation model will facilitate better performance at this stage. Also using the higher fitness individuals, chosen out of a larger set should give an initial boost to the evolutionary process.

Step Four: Rank the candidate solutions based on their fitness value.

Step Five: Preserve the elite by carrying over the best candidate solution to the next generation.

Step Six: Select parents using suitable selection operator and apply genetic operators namely recombination and mutation to create children (new candidate solutions) for the next generation.

Step Seven: The SVM regression models created in Step two are applied to estimate the fitness of the children (new candidate solutions) created in Step six. This involves assignment of most likely or appropriate models to each candidate solution. The details of this prediction mechanism are given in Section IV.

Step Eight: The set of newly created candidate solutions is ranked based on their approximate fitness values.

Step Nine: The best performing newly created candidate solution and the elite selected in Step five are carried to the population of the next generation.

Step Ten: New candidate solutions or children are created as described in Step six.

Step Eleven: Repeat Step seven to Step ten until either of the following condition is reached:

i. The predetermined maximum number of generations has been reached; or

ii. The periodic retraining of the SVM regression models is due.

Step Twelve: If the periodic retraining of the SVM regression models is due, this will involve actual evaluation of the candidate solutions in the current population. Based on this training data new regression models are formed. The algorithm then proceeds to execute Step four to Step eleven.

Remarks: The idea behind using periodic retraining of the SVM regression models is to ensure that the models continue to be representatives of the progressive search areas in the solution space.

V. Functional Aspects of DAFHEA-II

The following sections detail the major functional aspects of the DAFHEA-II framework.

A. Single Approximation Model Formation Using SVM Regression

The theoretical background of support vector machine is mainly inspired from statistical learning theory [24]. Major advantages of the support vector machines over other machine learning models such as neural networks, are that there is no local minima during learning and the generalization error does not depend on the dimension of the

space. Also the fast learning ability of the SVM regression [4, 2] model is a desirable property for on-line learning. In DAFHEA (both original and enhanced versions), as the approximation model has to be rebuilt frequently to be representative of the progressing solution space, this is an important criterion for model selection.

Let us consider the problem of approximating the set of data,

$$D = \{(x^1, y^1),...,(x^l, y^l)\} \tag{16}$$

with a linear function,

$$f(\mathbf{x}) = \mathbf{w}.\mathbf{x} + b,\ \mathbf{w}, \mathbf{x} \in R^n, b \in R \tag{17}$$

The construction of a model is reduced to the minimization of the following regularized ε-insensitive loss function:

$$L = \|\mathbf{w}\|^2 + C.\frac{1}{l}\sum_{i=1}^{l} \max\{|y_i - f(\mathbf{x}_i)| - \varepsilon\} \tag{18}$$

where ε is the tolerable error, C is a pre-specified regularization constant and f is the function to be estimated.

The minimization of (18) is equivalent to the following constrained optimization problem, giving the optimal regression function as:

$$\min \frac{1}{2}\|\mathbf{w}\|^2 + C \cdot \frac{1}{l}\sum_{i=1}^{l}(\xi_i + \xi_i^*) \tag{19}$$

subject to $((\mathbf{w} \cdot \mathbf{x}_i) + b) - y_i \le \varepsilon + \xi_i$ (20)

$$y_i - ((\mathbf{w}.\mathbf{x}_i) + b) \le \varepsilon + \xi_i^* \tag{21}$$

$$\xi_i, \xi_i^* \ge 0,\ i = 1,...,l \tag{22}$$

where ξ_i and ξ_i^* are slack variables representing upper and lower constraints on the output of the system.

Thus, quadratic-programming techniques can be applied to solve the minimization problem.

In the enhanced version of DAFHEA (DAFHEA-II), a multiple model regression approach is used. The technique used closely follows the approach described in [21].

B. Multiple Model Regression

The multiple model regression involves the following two stages:

i. The *training/learning phase*, involving creation of the models based on training data.

ii. The *prediction phase*, involving assignment of the most likely model to each candidate data and estimation of improved response using the selected model.

1) The Training/Learning Phase

Let us consider a finite number of samples or training data $(\mathbf{x}_i, y_i), (i = 1,...,n)$ [21]. The learning involves two objectives:

(a) to estimate N target models from a set of possible models:

$$f_m(\mathbf{x}, \omega_m), (\omega_m \in \mathrm{E}_m, m = 1,...N) \tag{23}$$

Where E_m is a parametric space for model *m*. Each model estimate approximates the corresponding target model $f_m(\mathbf{x}, \omega_m^*) \to tr_m(\mathbf{x})$.

(b) to partition available training data set into N subsets, where each subset belong to an appropriate model. The input (**x**) and/or output (y) space will be thus partitioned into N disjoint regions.

It is clear from the above discussion that the creation of multiple models here can be viewed as a generalization of the traditional single-model estimation. Traditional regression is applied to estimate appropriate regression-like models in a progressive manner while partitioning the data set into subsets at the same time.

2) The Prediction Phase

Using a single model approach, estimating a response $\hat{y}$ for a given test input $\mathbf{x}$, simply amounts to deducing $\hat{y} = f(\mathbf{x}, \omega^*)$, where $f(\mathbf{x}, \omega^*)$ is a model predetermined based on the training data. In case of multiple model estimation, first an appropriate model has to be selected for the test input $\mathbf{x}$ and then the response $\hat{y}$ can be computed as $\hat{y} = f_c(\mathbf{x}, \omega^*)$, where $f_c(\mathbf{x}, \omega^*)$ is the specifically

chosen model for $\mathbf{x}$. However, it is not possible to select a model using $\mathbf{x}$ alone, as there may be overlapping of input domains for different models. Thus, selection of model should be based on the $(\mathbf{x}, y)$ values of the test data as described in [21]. For a given sample of test data $z = (\mathbf{x}, y)$ generated by an unknown model u and a set of models estimated during training stage:

$$f_i(\mathbf{x}, \omega_i^*), \mathbf{x} \in X_i (i = 1,...N) \tag{24}$$

to determine the appropriate or most likely model the *distance* between the test sample and each of the models in (24), has to be computed. Each model in (24) is defined as a region in the input (x) space and the mapping $f_c : \mathbf{x} \to y$ in this region. Therefore, the *distance* may be defined in the input (x) space or in the y-space, or some combination of the two.

3) Exploiting Approximate Knowledge to Avoid Premature Convergence

The basic idea is to encourage exploration along with exploitation in the initial generations. The underlying notion is to achieve this by using the concept of merit functions. The merit function $f_m(x)$ is conceptually similar to the one used in [14].

VI. Experiments And Discussions

We have tested the proposed algorithms on three popular benchmark test functions and their *noisy* versions. These are Spherical, Rosenbrock and Rastrigin (see Table I) functions. These benchmark functions in the test suit are scalable and are commonly used to assess the performance of optimization algorithms [12]. For all functions except Rosenbrock the global minimum is $f(x)=0$ at $\{x_i\}^n = 0$. Rosenbrock has a global minimum of $f(x)=0$ at $\{x_i\}^n = 1$.

The *noisy versions* of the above set of functions are defined as:

$$f_{Noisy}(\vec{x}) = f(\vec{x}) + N(\mu,\sigma^2) \quad (25)$$

where, $N(\mu,\sigma^2)$= Standard Normal (or Gaussian) distribution with mean, μ = 0 and variance, σ^2 = 1. The probability density function $f(x;\mu,\sigma^2)$ is given as below,

$$f(x;\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \quad (26)$$

All simulations were carried out using the following experiment setup: The population size of $10n$ was used for all the simulations, where n is the number of variables for the problem; for comparison purposes three sets of input dimensions are considered; namely, $n = 5, 10$ and 20. For all three cases, tenfold validation was done with the number of iterations being 1000 for all non-noisy versions of the test problems; the SVM regression models were trained with *five times* the real EA (GA in this case) population size initially. However, incase of the noisy versions of the test functions much larger number of iterations have been used to obtain acceptable level of accuracy of results. All the simulation processes were executed using a Pentium® 4, 2.4GHz CPU processor. Tables II, III and IV show the comparative results of the various simulations runs using canonical GA model which uses only actual function evaluations and the proposed DAFHEA and DAFHEA-II models which use actual function evaluations sparingly. We report the results for the 5-D (dimension), 10-D (dimension) and 20-D (dimension) scenarios for both non-noisy and the noisy versions of the test functions. The reported results were obtained by achieving same level of tolerance for both canonical GA and the proposed models. For comparison purpose, we have used results reported in [12] (see Table II, III and IV).

Due to space constraints the function values (mean +

TABLE I
List of Test Benchmark Functions

Function	Formula
Spherical	$f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$
Rosenbrock	$f(\mathbf{x}) = \sum_{i=1}^{n-1} (x_i - 1)^2 + 100(x_i^2 - x_{i+1})^2$
Rastrigin	$f(\mathbf{x}) = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i))$

TABLE II
Total Function Evaluations Required (5-Dimensional Scenario)

Function	Canonical GA (Actual function evaluations)	DAFHEA Framework (Actual function evaluations)	DAFHEA-II Framework (Actual function evaluations)
Spherical	49045	21210	21200
Spherical (Noisy)	100,000	70000	91000
Rosenbrock	18000	7015	7009
Rosenbrock (Noisy)	100,000	71000	95600
Rastrigin	16500	4550	4545
Rastrigin (Noisy)	100,000	70000	93000

TABLE III
Total Function Evaluations Required (10-Dimensional Scenario)

Function	Canonical GA (Actual function evaluations)	DAFHEA Framework (Actual function evaluations)	Proposed DAFHEA-II Framework (Actual function evaluations)
Spherical	99150	77520	77500
Spherical (Noisy)	100,000	71000	92000
Rosenbrock	16500	6990	6985
Rosenbrock (Noisy)	100,000	71250	95482
Rastrigin	17100	7175	7175
Rastrigin (Noisy)	100,000	70000	93450

TABLE IV
Total Function Evaluations Required (20-Dimensional Scenario)

Function	Canonical GA (Actual function evaluations)	DAFHEA Framework (Actual function evaluations)	Proposed DAFHEA-II Framework (Actual function evaluations)
Spherical	199200	110420	110400
Spherical (Noisy)	500,000	410,000	470,000
Rosenbrock	70447	21170	21150
Rosenbrock (Noisy)	500,000	400,000	450,000
Rastrigin	101650	28010	28000
Rastrigin (Noisy)	500,000	410,500	460,000

standard error) could not be included here. Naturally there will be a tradeoff between precision of obtained result and number of function evaluations specifically in case of the noisy versions of the problems. However, it is obvious that the proposed DAFHEA and DAFHEA-II models effectively reduce the number of actual evaluations for all the benchmark function in the test suit. Formation and maintenance of the regression models incorporates additional computational expense. However, we need to remember that this approximation based evolutionary algorithm model is not proposed for regular optimization problems where actual function evaluation is not a matter of concern.

VII. CONCLUSIONS

Application of population based, iterative techniques like EA to expensive optimization problem domains is realistically feasible only if the number of actual function evaluations can be kept to a minimum. This can be achieved by the use of approximation or surrogate models to replace actual functions. In this paper brief description of tools to generate such meta models has been outlined. Evolutionary algorithm frameworks that replace actual function evaluation by support vector machine regression tool generated meta model evaluation, have been presented. In this framework a multiple model approach for support vector machine regression is used to develop the approximate models. The algorithms showed reliable performance in terms of accuracy for both noisy and non-noisy versions of commonly used benchmark problems. The overhead cost towards developing and maintaining the meta-model is not alarmingly high. Since this overhead is expected not to increase much with increased problem complexity, both the versions of DAFHEA should lead to considerable speed up for complex real life problems. As mentioned earlier the DAFHEA-II framework is an enhancement of the original DAFHEA to extend its application to problems involving uncertain fitness functions. The satisfactory performance of DAFHEA-II in case of noisy functions validates our claim that the framework is suitable for solving complex real world optimization problems involving uncertain environments.

REFERENCES

[1] A. Ratle.., "*Accelerating the convergence of evolutionary algorithms by fitness landscape approximation*", Parallel Problem Solving from Nature-PPSN V, Springer-Verlag, pp. 87-96, 1998.

[2] A. Smola and B. Schölkopf, "*A Tutorial on Support Vector Regression*", NeuroCOLT Technical Report NC-TR-98-030, Royal Holloway College, University of London, UK, 1998.

[3] B. Dunham, D. Fridshal., R. Fridshal and J. North, "*Design by natural selection*", Synthese, 15, pp. 254-259, 1963.

[4] B. Schölkopf , J. Burges and A. Smola, ed., "*Advances in Kernel Methods: Support Vector Machines*", MIT Press, 1999.

[5] C. Bishop, "*Neural Networks for Pattern Recognition*", Oxford Press, 1995.

[6] D. Büche., N. Schraudolph, and P. Koumoutsakos, "*Accelerating Evolutionary Algorithms Using Fitness Function Models*", Proc. Workshops Genetic and Evolutionary Computation Conference, Chicago, 2003.

[7] H. D. Vekeria and i. C. Parmee, "*The use of a co-operative multi-level CHC GA for structural shape optimization*", Fourth European Congress on Intelligent Techniques and Soft Computing – EUFIT'96, 1996.

[8] H. S. Kim and S. B. Cho, " *An efficient genetic algorithm with less fitness evaluation by clustering*", Proceedings of IEEE Congress on Evolutionary Computation, pp. 887-894, 2001.

[9] J. Sacks, W. Welch, T. Mitchell and H. Wynn, "*Design and analysis of computer experiments*", Statistical Science, 4(4), 1989.

[10] K. Rasheed, "*An Incremental-Approximate-Clustering Approach for Developing Dynamic Reduced Models for Design Optimization*", Proceedings of IEEE Congress on Evolutionary Computation, 2000.

[11] K. Rasheed, S. Vattam and X. Ni., "*Comparison of Methods for Using Reduced Models to Speed Up Design Optimization*", The Genetic and Evolutionary Computation Conference (GECCO'2002), 2002.

[12] K. Won, T. Roy and K. Tai, "*A Framework for Optimization Using Approximate Functions*", Proceedings of the IEEE Congress on Evolutionary Computation' 2003, Vol.3, IEEE Catalogue No. 03TH8674C, ISBN 0-7803-7805-9.

[13] M. A. El-Beltagy and A. J. Keane, "*Evolutionary optimization for computationally expensive problems using Gaussian processes*", Proc. Int. Conf. on Artificial Intelligence (IC-AI'2001), CSREA Press, Las Vegas, pp. 708-714, 2001.

[14] M. Bhattacharya and G. Lu, "*DAFHEA: A Dynamic Approximate Fitness based Hybrid Evolutionary Algorithm*", Proceedings of the IEEE Congress on Evolutionary Computation' 2003, Vol.3, IEEE Catalogue No. 03TH8674C, ISBN 0-7803-7805-9, pp. 1879-1886.

[15] M. Bhattacharya, "*Surrogate based Evolutionary Algorithm for Engineering Design Optimization*", Proceedings of the Eighth International Conference on Cybernetics, Informatics and Systemic (ICCIS 2005), ISBN 975-98458-9-X, pp. 52-57.

[16] P. Hajela and A. Lee., "*Topological optimization of rotorcraft subfloor structures for crashworthiness considerations*", Computers and Structures, vol.64, pp. 65-76, 1997.

[17] R. Myers and D. Montgomery, "*Response Surface Methodology*", John Wiley & Sons, 1985.

[18] S. Pierret, "*Three-dimensional blade design by means of an artificial neural network and Navier-Stokes solver*", Proceedings of Fifth Conference on Parallel Problem Solving from Nature, Amsterdam, 1999.

[19] S. R. Gunn, "*Support Vector Machines for Classification and Regression*", Technical Report, School of Electronics and Computer Science, University of Southampton, (Southampton, U.K.), 1998.

[20] T. Hastie, R. Tibshirani, J. Friedman, "*The Elements of Statistical Learning: Data Mining, Inference, and Prediction*", Springer Series in Statistics, ISBN 0-387-95284-5.

[21] V. Cherkassky and Y. Ma, "Multiple Model Estimation: A New Formulation for Predictive Learning", under review in IEE Transaction on Neural Network.

[22] V. Torczon and M. W. Trosset, "*Using approximations to accelerate engineering design optimisation*", ICASE Report No. 98-33. Technical report, NASA Langley Research Center Hampton, VA 23681-2199, 1998.

[23] V. V. Toropov, a. A. Filatov and A. A. Polykin, "*Multiparameter structural optimization using FEM and multipoint explicit approximations*", Structural Optimization, vol. 6, pp. 7-14, 1993.

[24] V. Vapnik, "*The Nature of Statistical Learning Theory*", Springer-Verlag, NY, USA, 1999.

[25] Y. Jin, M. Olhofer and B. Sendhoff, "*A Framework for Evolutionary Optimization with Approximate Fitness Functions*", IEEE Transactions on Evolutionary Computation, 6(5), pp. 481-494, (ISSN: 1089-778X). 2002.

[26] Y. Jin, M. Olhofer and B. Sendhoff., "*On Evolutionary Optimisation with Approximate Fitness Functions*", Proceedings of the Genetic and Evolutionary Computation Conference GECCO, Las Vegas, Nevada, USA. pp. 786- 793, July 10-12, 2000.

[27] Y. Jin, *"A Comprehensive Survey of Fitness Approximation in Evolutionary Computation"*, Soft Computing, 9(1), pp.3-12, 2005.