Expensive Optimization, Uncertain Environment: An EA-Based Solution

Maumita Bhattacharya
SOBIT, Charles Sturt University
Australia - 2640
maumita.bhattacharya@ieee.org

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 any population based iterative technique such as evolutionary algorithm in such problem domains is thus practically prohibitive. A feasible alternative is to build surrogates or use an approximation of the actual fitness functions to be evaluated. Naturally these surrogate or meta models are order of magnitude cheaper to evaluate compared to the actual function evaluation. This paper presents two evolutionary algorithm frameworks which involve surrogate based fitness function evaluation. The first framework, namely the Dynamic Approximate Fitness based Hybrid EA (DAFHEA) model [1] 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 problem domains involving uncertain environment. The second model, DAFHEA-II, an enhanced version of the original DAFHEA framework, incorporates a multiple-model based learning approach for the support vector machine approximator to handle uncertain environment [2]. Empirical evaluation results have been presented based on application of the frameworks to commonly used benchmark functions.

Categories and Subject Descriptors

Computing Methodologies [I.2 **Artificial Intelligence**]: I.2.8 Problem Solving, Control Methods, and Search.

General Terms

Algorithms, Design, Performance.

Keywords

Premature convergence, evolutionary algorithm, approximation.

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

1. INTRODUCTION AND BACKGROUND

The optimization of complex, high dimensional, multimodal problems often poses a problem which in fact depends on the structure inherent in the problems. Examples of such problem domains include large-scale finite element analysis (FEA), computational fluid dynamics (CFD), engineering design problems etc. For example, optimization problems involving engineering design are often characterized by features that make them highly computationally expensive to be solved by evolutionary algorithms or even other standard nonlinear optimization techniques. One of the most critical of these features is that the functions used to define such optimization problems are often computationally intensive. In engineering design optimization problem domains, a design can be represented by a number of continuous design parameters and the potential solutions are vectors in a multidimensional vector space [11]. Determining the *fitness* of these potential solutions generally involve some form of simulation to compute the relevant physical properties of the *object* represented by the solution vector and consolidate them into a single measure of merit, including the information on the status of the constraints if necessary. 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. The computational expense of evaluating the functions that define the problem would necessarily be incurred for *each* iteration, within any *iterative* optimization algorithm and in a population-based search algorithm like EA, for the entire population or part of the population, as is required, in *each* iteration. 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 [14,17,18]. 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, it can not be denied that accuracy of the result is a major risk involved in using meta-models to replace actual function evaluation [24, 12, 13, and 11]. When it is infeasible to precisely judge when, where and how much of such replacement is optimal, using a controlled approach holds the answer.

The use of approximate model to speed up optimization dates all the way back to the sixties [1]. The most widely used models being Response Surface Methodology [15], Krieging models [20] and artificial neural network models [3]. 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 [11].

In the multidisciplinary optimization (MDO) community, primarily response surface analysis and polynomial fitting techniques are used to build the approximate models [9, 24]. These models work well when single point traditional gradient-based optimization methods are used. However, 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 [26]. As is observed in [11], 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 [17, 18], 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* [11]. Here, the population is clustered into several groups and true evaluation is done only for the cluster representative [14]. 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 [12, 13] analysed the convergence property of approximate fitness based evolutionary algorithm. It has been observed that incorrect convergence can occur due to false optima 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 [4]. 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 [19, 23, and 4]. Yet in another approach as proposed in [12], 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.

While dealing with complex real world optimization problems, expensive function evaluations can be feasibly used only in a limited manner. Research on using surrogate models should focus on:

1) Minimizing uncertainty in approximate estimation
2) Employing corrective measures
3) Exploring ways to exploit the approximate knowledge for improving the optimization technique

The hybrid evolutionary algorithm framework, DAFHEA (dynamic approximate fitness based hybrid evolutionary algorithm) addresses some of the above issues [1]. DAFHEA replaces expensive function evaluation by its support vector machine (SVM) approximation. The concept of *merit function* [23] 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 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 2 presents the basic frameworks of DAFHEA and DAFHEA-II (the enhanced version of DAFHEA). Experiment details and simulation results are presented in Section 3. Finally conclusions are drawn in Section 4.

2. THE PROPOSED FRAMEWORKS

The proposed DAFHEA framework 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 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

following section presents the basic algorithm structure of DAFHEA.

2.1 Algorithm Structure of DAFHEA

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 model using normalised expensive function evaluation values as training set for *off-line training*. (Use of normalised values in the training set appears to improve performance of *meta model*, reducing effects of unnaturally high or low values). SVM hyper-parameters are initially tuned based on this training set.

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: Select parents using suitable selection operator and apply genetic operators namely recombination and mutation to create new generation.

Step Five: Use SVM approximation model to compute fitness of new generation individuals based on approximate evaluation. Form m distance-based (*considering spatial distribution of individuals*) clusters in the new population space. If for some n clusters, the standard deviation $\sigma \geq$ *Predefined Threshold,* rearrange solution space into $m+n$ clusters. Compute a *merit function* $f_m(x)$ as below:

$$f_m(x) = f_a(x) - \rho_1\sigma_i - \rho_2 d_{ij} - \rho_3 s_i \qquad (1)$$

where, $f_a(x)$ is the predicted fitness function value. σ_i is standard deviation (*in terms of objective value*) for the i^{th} cluster and d_{ij} is the normalized *minimum* Euclidean distance of j^{th} point of i^{th} cluster from the all truly evaluated points so far [5]. s_i is the sparseness of the i^{th} cluster. ρ_1, ρ_2 and ρ_3 are scaling factors for σ_i, d_{ij} and s_i respectively.

$$s_i = \frac{No\ of\ individuals\ in\ cluster\ i}{Dimension\ of\ individual} \qquad (2)$$

Step Six: Dynamically update the approximate model as below:

1) Identify the cluster containing the optimum based on approximation.

2) Perform expensive evaluation for the approximate optimum and its $k-$nearest neighbours. Also perform expensive evaluation for the centroid of all other data clusters and their $k-$nearest neighbours. Expand neighbourhood for true evaluation until a point is found in each space dimension such that % error $\delta \leq$ *Predefined threshold*. Here,

$$\delta = \left| \frac{a_{it} - a_{ip}}{a_{it}} \right| \times 100 \qquad (3)$$

where, a_{it} =True value of the i^{th} neighbour and a_{ip} =Predicted value of the i^{th} neighbour and **max** $i = k$.

3) Add the newly evaluated points to approximate model training set to update model.

Step Seven: When termination/evolution control criteria are not met, repeat *Step Four* to *Step Seven*.

It must be noted, the optimum is considered based on the original predicted value $f_a(x)$. For all other purposes fitness based on the *merit function* $f_m(x)$ is considered. Periodic parameter tuning of the SVM approximation model was incorporated, though no specific criterion was used.

The following section presents the enhanced DAFHEA or DAFHEA-II framework that incorporates the enhancement to tackle situations where the inputs to the meta model generation tools are generated by different models.

2.2 Algorithm Structure of Enhanced DAFHEA

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.

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
    }
}
}
```

Figure 1. DAFHEA-II framework.

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

2.2.1 Implementation of DAFHEA-II Framework

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.

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.

2.2.1.1 Single Approximation Model Formation with SVM Regression

The theoretical background of support vector machine is mainly inspired from statistical learning theory [25]. 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 [21, 22] 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,

$$D = \left\{\left(x^1, y^1\right),...,\left(x^l, y^l\right)\right\} \tag{4}$$

with a linear function,

$$f(\mathbf{x}) = \mathbf{w}.\mathbf{x} + b\,,\ \mathbf{w},\mathbf{x} \in R^n, b \in R \tag{5}$$

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

$$L = \|\mathbf{w}\|^2 + C.\frac{1}{l}\sum_{i=1}^{l}\max\left\{\left|y_i - f(\mathbf{x}_i)\right| - \varepsilon\right\} \tag{6}$$

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

The minimization of (6) is equivalent to the following constrained optimization problem, giving the optimal regression function as:

$$\textbf{min}\ \frac{1}{2}\|\mathbf{w}\|^2 + C\cdot\frac{1}{l}\sum_{i=1}^{l}\left(\xi_i + \xi_i^*\right) \tag{7}$$

$$\text{subject to}\left(\left(\mathbf{w}\cdot\mathbf{x}_i\right)+b\right) - y_i \le \varepsilon + \xi_i \tag{8}$$

$$y_i - \left(\left(\mathbf{w}.\mathbf{x}_i\right)+b\right) \le \varepsilon + \xi_i^* \tag{9}$$

$$\xi_i, \xi_i^* \ge 0\ ,\ i = 1,...,l \tag{10}$$

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 [5].

2.2.1.2 Multiple Model Regression Technique

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.

2.2.1.2.1 The Training/Learning Phase

Let us consider a finite number of samples or training data $(\mathbf{x}_i, y_i), (i = 1,...,n)$ [5]. The learning involves two objectives:

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

$$f_m(\mathbf{x}, \omega_m), (\omega_m \in \mathrm{E}_m, m = 1,...N) \tag{11}$$

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.

2.2.1.2.2 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 [5]. 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:

$$f_i(\mathbf{x}, \omega_i^*), \mathbf{x} \in X_i (i = 1,...N) \tag{12}$$

to determine the appropriate or most likely model the *distance* between the test sample and each of the models in (12), has to be computed. Each model in (12) is defined as a region in the input (x) space and the mapping $f_c : \mathbf{x} \rightarrow y$ 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.

2.2.1.3 Use of Approximate Knowledge to Curb Premature Convergence

Please see [1] for details.

3. EXPERIMENTS

3.1 Method Used for Result Comparison

For comparison purposes we have used the results reported in [27]. Won et al. in [27] presents a population-based, stochastic, zero order, elite preserving algorithm that uses approximate function evaluation in lieu of actual function evaluations. Details of the method are given in [27]. Performances of DAFHEA and DAFHEA-II have been evaluated against that of Canonical GA and the proposed method of Won et al. as reported in [27].

3.2 Test Details and Discussions

The performance of the proposed algorithms is tested on five popular benchmark test functions (see Table 1): namely, Spherical, Rosenbrock, Rastrigin, Schwefel and Ellipsoidal. These benchmark functions in the test suit are scalable and are commonly used to assess the performance of optimization algorithms. For all five 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 $\{x_i\}^n = 1$.

All simulations were carried out using the following assumptions: 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 generations being 1000; the SVM regression models were trained with *five times* the real GA population size initially and all the simulation processes were executed using a Pentium® 4, 2.4GHz CPU processor. Tables 2, 3 and 4 show the comparative statistics 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. The reported results were obtained by achieving same level of tolerance for both canonical GA and the proposed models. For comparison purpose, results reported in [27] were considered (see Table 2, 3 and 4). The results reported for Canonical GA and Won et al.'s proposed method are averages of ten independent runs [27]. However, results reported for DAFHEA and DAFHEA-II are averages of thirty independent runs.

It is clear from the depicted results that the proposed DAFHEA and DAFHEA-II models effectively reduce the number of actual function evaluations for most of the benchmark function in our test suit to obtain similar level of solution accuracy. Due to space constrains the actual solutions could not be presented here. It is true that the formation and maintenance of the regression models incorporates additional computational expense. However, this approximation based evolutionary algorithm model is not proposed for regular optimization problems where actual function evaluation is not a matter of concern. Complex real world problems involving very expensive function evaluations will benefit from such approximation based algorithms even when the reduction in the number of actual evaluations is relatively quite low.

Table 1. Description of Benchmark Functions

Function	Formula
Spherical	$f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$
Ellipsoidal	$f(\mathbf{x}) = \sum_{i=1}^{n} i x_i^2$
Schwefel	$f(\mathbf{x}) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$
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 2. Total Evaluations required (5-Dimensional case) where, M_1=Canonical GA, M_2 = Proposed method as described in [27], M_3 = DAFHEA Framework, M_4 = DAFHEA-II Framework. All values shown here represent total number of actual function evaluations involved in the specific methods.

Function	M_1	M_2	M_3	M_4
Spherical	49045	21450	21210	21200
Ellipsoidal	49045	21051	21000	21000
Schwefel	49045	25951	25500	25500
Rosenbrock	18000	7201	7015	7009
Rastrigin	16500	4601	4550	4545

Table 3. Total Evaluations required (10-Dimensional case) where, M_1=Canonical GA, M_2 = Proposed method as described in [27], M_3 = DAFHEA Framework, M_4 = DAFHEA-II Framework. All values shown here represent total number of actual function evaluations involved in the specific methods.

Function	M_1	M_2	M_3	M_4
Spherical	99150	77567	77520	77500
Ellipsoidal	99150	84334	84310	84300
Schwefel	99150	53834	53755	53750
Rosenbrock	16500	7001	6990	6985
Rastrigin	17100	7100	7175	7075

Table 4. Total Evaluations required (20-Dimensional case) where, M_1=Canonical GA, M_2 = Proposed method as described in [27], M_3 = DAFHEA Framework, M_4 = DAFHEA-II Framework. All values shown here represent total number of actual function evaluations involved in the specific methods.

Function	M_1	M_2	M_3	M_4
Spherical	199200	110467	110420	110400
Ellipsoidal	199200	81534	81450	81450
Schwefel	199200	144267	144220	144200
Rosenbrock	70447	21201	21170	21150
Rastrigin	101650	28020	28010	27890

4. CONCLUSIONS

Population based, iterative optimization techniques such as evolutionary algorithms can not be feasibly used in problem domains that involve expensive fitness evaluation. Use of approximation or meta models to replace actual functions is an attractive choice to address this issue. This can drastically lower the computational expense of using EA to solve complex optimization problems. In this paper two evolutionary algorithm techniques that replace actual function evaluation by SVM regression tool generated meta model evaluation, have been presented. The second framework is an enhanced version of the first. In this, a multiple model approach for support vector machine regression is used to develop the approximate models. The algorithms showed reliable performance in terms of solution accuracy and reduction in actual function evaluations. 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 [2] framework is an enhancement of the original DAFHEA [1] to extend its application to problems involving uncertain fitness functions. The enhanced framework is suitable for solving complex real world optimization problems where the input information is expected to be generated by multiple models instead of a single model due to presence of noise. Our future research will investigate mechanisms to reduce the overhead cost of developing and maintaining the surrogate models.

5. REFERENCES

[1] Bhattacharya, M. and Lu, G. 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.

[2] Bhattacharya, M. 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.

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

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

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

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

[7] El-Beltagy, M. A. and Keane, A. J. 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.

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

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

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

[11] Jin, Y. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing Journal*, 9(1), Springer, pp. 3-12, 2005.

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

[13] Jin, Y., Olhofer, M. and Sendhoff, B. 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.

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

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

[16] Pierret, S. 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.

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

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

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

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

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

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

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

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

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

[26] Vekeria, H. D. and Parmee, I. C. 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.

[27] Won, K., Roy, T. and Tai, K. 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.