Reduced Computation for Evolutionary Optimization in Noisy Environment

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

ABSTRACT

Evolutionary Algorithms' (EAs') application to real world optimization problems often involves expensive fitness function evaluation. Naturally this has a crippling effect on the performance of any population based search technique such as EA. Estimating the fitness of individuals instead of actually evaluating them is a workable approach to deal with this situation. Optimization problems in real world often involve expensive fitness. In [14] and [15] we presented two EA models, namely DAFHEA (Dynamic Approximate Fitness based Hybrid Evolutionary Algorithm) and DAFHEA-II respectively. The original DAFHEA framework [14] reduces computation time by controlled use of meta-models generated by Support Vector Machine regression to partly replace actual fitness evaluation by estimation. DAFHEA-II [15] is an enhancement to the original framework in that it can be applied to problems that involve uncertainty. DAFHEA-II, incorporates a multiple-model based learning approach for the support vector machine approximator to filter out effects of noise [15]. In this paper we present further investigation on the performance of DAFHEA and DAFHEA-II.

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: Evolutionary algorithm, uncertainty, approximation.

1. INTRODUCTION

Many real world optimization problems involve very expensive function evaluation, making it impractical for a population based search technique such as EA to be used in such problem domains. 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. A suitable alternative is to use approximation instead of actual function evaluation to substantially reduce the computation time [8, 10, and 11]. Use of approximate model to speed up optimization dates all the way back to the sixties [14]. 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]. The most widely used meta-model generators are the Response Surface Methodology [17], the Krieging models [9] and the 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 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.

The DAFHEA (dynamic approximate fitness based hybrid evolutionary algorithm) framework proposed in our earlier research [14] replaces expensive function evaluation by its support vector

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.
ACM 978-1-60558-131-6/08/07.

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 [15] 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.

In our current research, we present an investigation on the performance of DAAFHEA-II framework as regards to its ability to reduce computation effort while dealing with expensive problem domains involving uncertainty.

The paper is organized as follows: The basic concepts and the implementation details of the *hybrid EA frameworks* DAFHEA-II are outlined in Section 2. Experiment results and discussions are presented in Section 3. Finally the conclusions are drawn in Section 4.

Table 1. Benchmark Test Functions Used in Simulation.

Function	Formula
Spherical	$f_{sph}(x) = \sum_{i=1}^{M} x_i^2$
Ellipsoidal	$f_{elp}(x) = \sum_{i=1}^{M} i x_i^2$
Schwefel	$f_{sch-1.2}(x) = \sum_{i=1}^{M} \left(\sum_{k=1}^{i} x_k \right)^2$
Rosenbrock	$f_{ros}(x) = \sum_{i=1}^{n-1} \left(100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2 \right)$
Rastrigin	$f_{rtg}(x) = \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) + 10 \right)$

2. THE APPROXIMATION BASED EA

As in the original DAFHEA framework [14], DAFHEA-II [15] 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 true 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.

2.1 Functional Details

The operational detail of DAFHEA-II [15] framework is as described 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*.

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.

3. EXPERIMENTS AND DISCUSSIONS

We have tested the proposed algorithms on a set of popular benchmark test functions and their *noisy* versions. These are Spherical, Ellipsoidal, Schwefel, Rosenbrock and Rastrigin (see Table 1) 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

Table 2. Results for Simulations of DAFHEA-II on Benchmark Test Functions (Non-Noisy).

		Best Fitness	Worst Fitness	Mean Fitness	Median Fitness	Standard Deviation
5D	$f_{sph}(x)$	**0.091E-60**	**1.69E-55**	**1.138E-56**	**0.932E-57**	**2.518E-56**
	$f_{elp}(x)$	**3.510E-53**	**1.71E-48**	**3.412E-51**	**1.131E-51**	**0.198E-50**
	$f_{sch-1.2}(x)$	**1.186E-50**	**3.215E-47**	**1.911E-48**	**3.391E-49**	**7.019E-48**
	$f_{ros}(x)$	1.198E-40	2.011E-37	1.998E-38	1.019E-38	**2.081E-38**
	$f_{rtg}(x)$	**1.413E-2**	**0.0109**	**3.322E-1**	**1.191E-1**	**4.114E-1**
10D	$f_{sph}(x)$	**1.543E-46**	**1.99E-42**	**1.588E-43**	**0.119E-43**	**0.932E-42**
	$f_{elp}(x)$	0.912E-41	1.33E-38	2.523E-39	1.038E-39	**4.042E-39**
	$f_{sch-1.2}(x)$	1.012E-40	1.88E-37	2.971E-38	1.891E-38	**3.237E-38**
	$f_{ros}(x)$	1.109E-28	2.190E-25	1.918E-26	1.001E-26	**2.097E-26**
	$f_{rtg}(x)$	**1.738E-1**	**1.998**	**3.388E-1**	**3.013E-1**	**4.133E-1**
20D	$f_{sph}(x)$	**1.118E-38**	**1.99E-34**	**1.388E-35**	**0.108E-35**	**2.032E-35**
	$f_{elp}(x)$	1.112E-34	1.72E-31	1.323E-32	1.11E-32	**2.838E-32**
	$f_{sch-1.2}(x)$	1.011E-32	1.75E-30	1.989E-31	1.001E-31	**2.136E-31**
	$f_{ros}(x)$	1.110E-21	2.101E-18	1.901E-19	1.001E-19	**1.996E-19**
	$f_{rtg}(x)$	**5.012**	**20.113**	**10.032**	**11.192**	**7.732**

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:

$$f_{Noisy}(\vec{x}) = f(\vec{x}) + N(\mu,\sigma^2) \quad (1)$$

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

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

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 for the results

reported in Table 2; 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. Table 2 presents the results obtained with DAFHEA-II algorithm on non-noisy versions of the benchmark test problems. Tables 3 presents the comparative results (number of actual function evaluations) 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. The results reported in [12] for a computation effort reduction technique has also been included for comparison purpose. Please note that, unlike the results reported in Table 2, these results were obtained by running the algorithms for variable number of iterations with the goal of achieving certain level of tolerance in the results. 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 DAFHEA and DAFHEA-II models. For comparison purpose, we have used results reported in [12]. While it is hard to generalize performance of the algorithms based on a small set of test functions, it is obvious from the simulation results that both DAFHEA and DAFHEA-II performs relatively efficiently as compared to the other techniques reported here.

4. 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, we have investigated the performance of an approximation based evolutionary algorithm, namely DAFHEA-II [15] as regards to its applicability to both noisy and non-noisy optimization problems. Performance of an earlier version of DAFHEA-II, namely DAFHEA has also been included. The algorithms showed reliable performance in terms of accuracy for both noisy and non-noisy versions of commonly used benchmark problems in majority of the test cases. 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. The satisfactory performance of DAFHEA-II in case of the noisy versions of the test functions validates our claim that the framework is suitable for solving complex real world optimization problems involving uncertain environments.

5. 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 EA for Expensive Optimization Problem*", Proceedings of the 2007 IEEE Congress on Evolutionary Computation (CEC 2007), Singapore, 1-4244-1340-0, 2007 IEEE Press.

[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*",

[19] Proceedings of Fifth Conference on Parallel Problem Solving from Nature, Amsterdam, 1999.

[20] 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.

[21] 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.

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

[23] 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.

[24] 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.

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

[26] 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.

[27] 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.

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

Table 3. Total Number of Actual Function Evaluations Required.

		Canonical GA	**DAFHEA**	**DAFHEA-II**	**Method described in [12]**
5D	$f_{sph}(x)$	49045	21210	21300	21450
	$f_{sph}(x)$ **(noisy)**	100,000	59000	58000	-
	$f_{elp}(x)$	49045	21000	21200	21051
	$f_{elp}(x)$ **(noisy)**	100,000	59000	58000	-
	$f_{sch-1.2}(x)$	49045	25500	26000	25951
	$f_{sch-1.2}(x)$ **(noisy)**	100,000	69000	68000	-
	$f_{ros}(x)$	18000	7015	7100	7201
	$f_{ros}(x)$ **(noisy)**	35,000	9500	9000	-
	$f_{rtg}(x)$	16500	4550	4570	4601
	$f_{rtg}(x)$ **(noisy)**	100,000	5500	5100	-
10D	$f_{sph}(x)$	99150	77520	77535	77567
	$f_{sph}(x)$ **(noisy)**	100,000	76000	75000	-
	$f_{elp}(x)$	99150	84310	84325	84334
	$f_{elp}(x)$ **(noisy)**	100,000	85000	84500	-

	$f_{sch-1.2}(x)$	99150	53755	53800	53834
	$f_{sch-1.2}(x)$ **(noisy)**	100,000	65000	64500	-
	$f_{ros}(x)$	16500	6990	6999	7001
	$f_{ros}(x)$ **(noisy)**	100,000	71250	71000	-
	$f_{rtg}(x)$	17100	**7175**	**7180**	7100
	$f_{rtg}(x)$ **(noisy)**	100,000	20500	20000	-
20D	$f_{sph}(x)$	199200	110420	110430	110467
	$f_{sph}(x)$ **(noisy)**	500,000	300,500	300,000	-
	$f_{elp}(x)$	199200	81450	81480	81534
	$f_{elp}(x)$ **(noisy)**	250,000	81550	81500	-
	$f_{sch-1.2}(x)$	199200	144220	144235	144267
	$f_{sch-1.2}(x)$ **(noisy)**	300,000	200,050	200,000	-
	$f_{ros}(x)$	70447	21170	21180	21201
	$f_{ros}(x)$ **(noisy)**	500,000	290,500	290,000	-
	$f_{rtg}(x)$	101650	28010	28090	28020
	$f_{rtg}(x)$ **(noisy)**	500,000	410,500	410,000	-