Evolutionary Optimization for Computationally expensive problems using Gaussian Processes

Mohammed A. El-Beltagy
Computational Engineering and Design Centre
University of Southampton
Highfield, Southampton SO17 1BJ
United Kingdom
mohammed@computer.org

Andy J. Keane
Computational Engineering and Design Centre
University of Southampton
Highfield, Southampton SO17 1BJ
United Kingdom
Andy.Keane@soton.ac.uk

Abstract *The use of statistical models to approximate detailed analysis codes for evolutionary optimization has attracted some attention [1-3]. However, those early methodologies do suffer from some limitations, the most serious of which being the extra tuning parameter introduceds. Also the question of when to include more data points to the approximation model during the search remains unresolved. Those limitations might seriously impede their successful application. We present here an approach that makes use of the extra information provided by a Gaussian processes (GP) approximation model to guide the crucial model update step. We present here the advantages of using GP over other neural-net biologically inspired approaches. Results are presented for a real world-engineering problem involving the structural optimization of a satellite boom.*

Keywords: Evolutionally Computation, Optimization, Gaussian Processes, Computationally expensive problems.

1 Introduction

The optimization of complex high dimensional, multimodal problems often requires a relatively high number of function evaluations. In many real world problems, this computational burden cannot be afforded. Examples of such problems include large-scale finite element analysis (FEA) or computational fluid dynamics (CFD) simulations. In such problems, the cost of a single function evaluation is in the order of hours of supercomputer time.

In is been proven useful to build approximate models of the expensive analysis code and use it for the purpose of carrying out optimization [4]. These approximate models are orders of magnitude cheaper to run than the full analysis codes. Many regression and interpolation tools could be used to construct such an approximation (e.g. least square regression, back propagating artificial neural net, response surface models, etc.).

In the multidisciplinary optimization (MDO) community the main focus has been on using response surface analysis and polynomial fitting techniques to build the approximate models. These methods have been proven to work well when single point traditional gradient-based optimization methods are used. However, they cannot cope with large dimensional multimodal problems since they generally carry out approximation using simple quadratic models. [5-11].

During the optimization, it is only feasible that the expensive analysis be used sparingly. This requirement clashes with that of good approximate model building, where as many sampled points are needed to obtain a good approximation. There is hence a need to develop a framework that balances those two requirements. Alexandrov *et al* [12] presented an approximation management framework for conducting gradient-based search.

In our earlier work [3], we presented a framework for approximate model update based on fitness and design of experiments (DOE) criteria. Unlike Ratle's work [1, 2], where only the most recently evaluated points were used to construct the approximation, in our approach the approximation model was continuously expanding; retaining all useful information provided by the evolutionary search. To limit the cost of model reconstruction, Ratle discarded most of the accurately evaluated points while updating his approximate model. It is our view that for problems of sufficient complexity and computational expense, the risk involved is discarding potentially useful points from the approximate model far outweigh the computational savings gained by discarding them.

Like Ratle, a major drawback of our approach is that there were up to five extra tuning parameters to adjust for model construction and update. In this paper a present a far more elegant and flexible approach for carrying approximate model update.

We use here a Gaussian process approximation model, since it has several attractive features. Most important of which is the ability to provide an error bar for each prediction. An algorithm for

approximate model update is built around this capability. Results on high dimensional multimodal real world engineering problem are quite encouraging.

This paper is organized as follows: in the next section we justify our choice of using Gaussian process over other NN techniques. In section 3 we provide a brief description of how prediction is carried out using Gaussian processes. Section 4 describes the Evolutionary-Gaussian processes synthesis. Section 5 describes the real world problem and details experimental results. The paper closes with a brief conclusion and discussion of future work.

2 Why Gaussian Processes?

Although other artificial neural network models could have been used (e.g. multilayer preceptron networks (MLP), and radial basis function networks RBFN), to construct an approximate model, Gaussian Process was preferred for the following reasons:

1. The ability to overcome the problem of over-fitting, given a limited data set: Over-fitting could be handled using other ANNs (by using regularization for instance), but still the solution obtained might be sensitive to the ANN's architecture (number of layers, number of nodes in each layer, …etc). It has been shown by Neal [14] that the properties of a neural network with one hidden layer converge to those of a Gaussian process as the number of hidden neurons tends to infinity if standard regularization "weight decay" priors are assumed.
2. A limited number of meaningful tunable parameters: Unlike the weights of other ANNs, the GP's hyperparameters are limited to the problem size. These optimized hyperparameters can indicate different functional relationships between the model parameters. Furthermore, if the model builder understands these functional relationships, s/he could incorporate this knowledge in the form of priors to be used during the optimization of the hyperparameters. This is expressed in the $P(\theta)$ term in equations (12) and (13).
3. Online data addition without the need of having to re-optimize the model parameters: This may substantially reduce the computational cost of increasing the number of training points after the GP has been optimized for a large subset of those points.

3 Prediction with Gaussian Processes

Given a training data set $\mathcal{D}$ consisting of N pairs of vector inputs $\mathbf{x}_n$ and scalar outputs t_n for n=1…N, a Gaussian process (GP) prediction model is concerned with evaluating the probability $P(t_{N+1}|\mathcal{D}, \mathbf{x}_{N+1})$. The input vector for a point to be predicted is denoted $\mathbf{x}_{N+1}$ and t_{N+1} is the corresponding prediction. The inputs are L-dimensional and the targets are scalar. $P(\mathbf{t}_N|\{\mathbf{x}_N\})$ is assumed to follow a Gaussian distribution given by

$$P(\mathbf{t}_N|\mathcal{D},\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^N|\mathbf{C}_N|}}\exp\left[-\frac{1}{2}(\mathbf{t}-\mu)^T\mathbf{C}_N^{-1}(\mathbf{t}-\mu)\right], \quad (1)$$

where $\mathbf{C}_N$ is the covariance matrix for $P(\mathbf{t}_N|\mathcal{D},\mathbf{x})$, and μ is the mean. $\{\mathbf{x}_N\}$ and $\mathbf{t}_N$ are the sets of all inputs and outputs in $\mathcal{D}$ respectively. For properly normalized data, it can be assumed that μ=0. The joint distribution for the training outputs and the prediction $\mathbf{t}_{N+1}$ will take a similar form to (1) and is given by

$$P(\mathbf{t}_{N+1}|\mathcal{D},\mathbf{x}_{N+1}) = \frac{1}{\sqrt{(2\pi)^{N+1}|\mathbf{C}_{N+1}|}}\exp\left[-\frac{1}{2}\mathbf{t}_{N+1}^T\mathbf{C}_{N+1}^{-1}\mathbf{t}_{N+1}\right], \quad (2)$$

where $\mathbf{t}_{N+1}=\{\mathbf{t}_N, t_{N+1}\}$. The predictive probability distribution for the prediction t_{N+1} is therefore

$$\begin{aligned} P(t_{N+1}|\mathcal{D},\mathbf{x}_{N+1}) &= \frac{P(\mathbf{t}_{N+1}|\mathcal{D},\mathbf{x}_{N+1})}{P(\mathbf{t}_N|\{\mathbf{x}_N\})} \\ &= \frac{1}{\sqrt{(2\pi)\frac{|\mathbf{C}_{N+1}|}{|\mathbf{C}_N|}}}\exp\left[-\frac{1}{2}(\mathbf{t}_{N+1}^T\mathbf{C}_{N+1}^{-1}\mathbf{t}_{N+1}-\mathbf{t}_N^T\mathbf{C}_N^{-1}\mathbf{t}_N)\right]. \end{aligned} \quad (3)$$

Note that the covariance matrix $\mathbf{C}_{N+1}$ is just $\mathbf{C}_N$ with an extra row and column attached:

$$\mathbf{C}_{N+1} = \left[\begin{array}{c|c} \mathbf{C}_N & \mathbf{k} \\ \hline \mathbf{k}^T & \kappa \end{array}\right], \quad (4)$$

where the $\mathbf{k}$ and the κ scalar are defined as

$$\mathbf{k}^T = [C(\mathbf{x}_1,\mathbf{x}_{N+1}), C(\mathbf{x}_2,\mathbf{x}_{N+1}), \cdots, C(\mathbf{x}_N,\mathbf{x}_{N+1})] \quad (5)$$

$$\kappa = C(\mathbf{x}_{N+1},\mathbf{x}_{N+1}). \quad (6)$$

Collecting the terms that are a function of t_{N+1} in equation (3), the Gaussian distribution could be expressed as:

$$P(t_{N+1}|\mathcal{D},\mathbf{x}_{N+1})=\frac{1}{\sqrt{(2\pi)\dfrac{|\mathbf{C}_{N+1}|}{|\mathbf{C}_N|}}}\exp\left[-\frac{(t_{N+1}-\hat{t}_{N+1})^2}{2\sigma^2_{\hat{t}_{N+1}}}\right], \quad (7)$$

where

$$\hat{t}_{N+1}=\mathbf{k}^T\mathbf{C}_N^{-1}\mathbf{t}_N, \quad (8)$$

$$\sigma^2_{\hat{t}_{N+1}}=\kappa-\mathbf{k}^T\mathbf{C}_N^{-1}\mathbf{t}_N. \quad (9)$$

Hence, the prediction is given by the mean value $\hat{t}_{N+1}$, and $\sigma^2_{\hat{t}_{N+1}}$ provides a measure of the confidence in the prediction. The error bars on the prediction are $\pm\sigma_{\hat{t}_{N+1}}$.

Elements of the covariance matrix $\mathbf{C}_N$ as well as in the above two expressions are calculated using the covariance function $C(\mathbf{x}_i,\mathbf{x}_j)$, thus $(\mathbf{C}_N)_{ij}=C(\mathbf{x}_i,\mathbf{x}_j)$. The covariance function used here is quite popular in the literature for the interpretability of its hyperparameters. It is given by

$$C(\mathbf{x}_i,\mathbf{x}_j)=\theta_1\exp\left[-\frac{1}{2}\sum_{l=1}^{L}\frac{\left(x_i^{(l)}-x_j^{(l)}\right)}{r_l^2}\right]+\theta_2+\delta_{ij}\theta_3 \quad (10)$$

here $x_n^{(l)}$ is the l^{th} component of $\mathbf{x}_n$. The hyperparameters are defined as $\theta=\log(\theta_1,\theta_2,\theta_3,\mathbf{r})$ and they have the following meanings: θ_1 controls the overall vertical scale relative to the mean of the Gaussian process, θ_2 sets the bias of the correlation, θ_3 sets the noise level, and $\mathbf{r}$ allows a different distance measure for each input dimension. For irrelevant inputs, the corresponding r_l will be large and the model will ignore that input. This property of $\mathbf{r}$ in GP is termed automatic relevance determination (ARD), and could prove very useful for understating the relevance of the various inputs as well as specifying priors during hyperparameter optimization. The hyperparameters are defined as the log of the variables in equation (10) to restrict their values to be positive.

Using Bayes' theorem, the posterior probability of the hyperparameters given the training data is

$$P(\theta|\mathcal{D})=\frac{P\left(\mathbf{t}_N|\{\mathbf{x}_N\},\theta\right)P\left(\theta|\{\mathbf{x}_N\}\right)}{P\left(\mathbf{t}_N|\{\mathbf{x}_N\}\right)} \quad (11)$$

Rather than maximizing (11) directly to determine the maximum *a posteriori* estimate for θ, the logarithm of the probability is maximized. The denominator of (11) is not a function of θ and will therefore be considered constant for optimization purposes. The logarithm of the posterior probability is hence

$$\begin{aligned}\mathcal{L}&=\ln P\left(\mathbf{t}_N|\{\mathbf{x}_N\},\theta\right)+\ln P\left(\theta|\{\mathbf{x}_N\}\right)-\ln P\left(\mathbf{t}_N|\{\mathbf{x}_N\}\right)\\&=\ln P\left(\mathbf{t}_N|\{\mathbf{x}_N\},\theta\right)+\ln P\left(\theta\right)+const\\&=-\frac{1}{2}\log|\mathbf{C}_N|-\frac{1}{2}\mathbf{t}_N^T\mathbf{C}_N^{-1}\mathbf{t}_N-\frac{N}{2}\log 2\pi+\ln P(\theta)+const.\end{aligned} \quad (12)$$

This maximization is usually carried out using a gradient-based optimizer. For the results presented here a conjugate gradient optimizer was used. The derivative of (12) with respect to one of the hyperparameters, θ is

$$\frac{\partial\mathcal{L}}{\partial\theta}=-\frac{1}{2}Tr\left(\mathbf{C}_N^{-1}\frac{\partial\mathbf{C}_N}{\partial\theta}\right)+\frac{1}{2}\mathbf{t}_N^T\mathbf{C}_N^{-1}\frac{\partial\mathbf{C}_N}{\partial\theta}\mathbf{C}_N^{-1}\mathbf{t}_N+\frac{\ln P(\theta)}{\partial\theta} \quad (13)$$

To summarize, the procedure is to first use equations (12) and (13) to obtain the *maximum a posteriori* value of θ. The hyperparameters obtained are then used to construct the final covariance matrix. This is then inverted and used with equations (5) and (6) to evaluate equations (8) and (9).

Since the focus here is on deterministic models, the noise level term θ_3 in equation (10) is not considered in covariance calculations.

3.1 Online model expansion with GP

The GP model can expand to include new data points with minimal computational cost. This is based on the assumption that the newly added points are not likely imply a significant change in the hyperparameters, i.e., that the general form of the function described by the new larger data set is approximately the same as the one described by the old. The new inverse of the covariance matrix $\tilde{\mathbf{C}}_L^{-1}$ can be constructed from the old $\mathbf{C}_N^{-1}$ using inversion by partitioning [13: 77-78]. The partitioned inverse equation is

$$\tilde{\mathbf{C}}_L^{-1}=\begin{bmatrix}\mathbf{M} & \hat{\mathbf{K}}\\ \hat{\mathbf{K}}^T & \hat{\mathbf{V}}\end{bmatrix}, \quad (14)$$

where

$$\hat{\mathbf{V}}=\left(\mathbf{V}-\mathbf{K}^T\mathbf{C}_N^{-1}\mathbf{K}\right)^{-1}, \quad (15)$$

$$\hat{\mathbf{K}}=-\mathbf{C}_N^{-1}\mathbf{K}\hat{\mathbf{V}}, \quad (16)$$

$$\mathbf{M}=\mathbf{C}_N^{-1}+\hat{\mathbf{K}}\mathbf{K}^T\mathbf{C}_N^{-1}, \quad (17)$$

and

$$\mathbf{K} = \begin{bmatrix} C(\mathbf{x}_1, \mathbf{x}_{N+1}) & \cdots & C(\mathbf{x}_1, \mathbf{x}_{N+M}) \\ \vdots & \ddots & \vdots \\ C(\mathbf{x}_N, \mathbf{x}_{N+1}) & \cdots & C(\mathbf{x}_N, \mathbf{x}_{N+M}) \end{bmatrix} \quad (18)$$

$$\mathbf{V} = \begin{bmatrix} C(\mathbf{x}_{N+1}, \mathbf{x}_{N+1}) & \cdots & C(\mathbf{x}_{N+1}, \mathbf{x}_{N+M}) \\ \vdots & \ddots & \vdots \\ C(\mathbf{x}_{N+1}, \mathbf{x}_{N+M}) & \cdots & C(\mathbf{x}_{N+M}, \mathbf{x}_{N+M}) \end{bmatrix}. \quad (19)$$

In the above expressions the number of newly added points is *M*. The size of the new inverse covariance matrix is *L*×*L*, where *L*=*N*+*M*. **K** is an *N*×*M* matrix, having the covariance of the newly added points to those that were already used for training. **V** is an *M*×*M* symmetric matrix comprising of the covariance of newly added points to one another.

After $\tilde{\mathbf{C}}_L^{-1}$ is constructed, it is used in equations (8) and (9) to carry out prediction and error estimation.

4 Evolutionary-Gaussian processes synthesis

Although the previous fitness and DOE based approach [3] have worked reasonably well, it contains many adjustable parameter that could be very much problem dependant (*generation delay*, *fitfac*, *doefac*, η, and ε). The approach presented here is simpler and more elegant. Instead of making distance calculations and attempting to satisfy design of experiments requirement, it simply makes use of the GP-predicted standard deviation as shown in Figure 1.

```
Input maxeval, maxstdtol
Begin
  Random population initialization
  Evaluation of Np individuals
  Nacc=Np
  OldNacc=Nacc
  Construct initial Gaussian process
  while(Nacc <  maxeval)
    Apply evolutionary operators
    for i=1 to Np
       stdtol = maxstdtol (maxeval − Nacc)/(maxeval − Np)
      if( σ(p_i) > stdtol )
       evaluate p_i using expensive model
       expand the GP to include p_i
       Nacc=Nacc+1
      end if
      else
       evaluate p_i using the GP
      end else
    end for
    if(OldNacc==Nacc)
      maxstdtol=maxstdtol/2
    end if
    OldNacc=Nacc
  end while
End
```

Figure 1 Pseudo code for the evolutionary-GP synthesis

In the above pseudo code, the population consists of *Np* individuals. The parameters for each population member is denoted by $\mathbf{p}_i$, where *i*=1… *Np*. The maximum number of affordable expensive model calculation is *maxeval*. *maxstdtol* is a parameter that describes the maximum allowable tolerance on the prediction uncertainty.

The standard deviation on the model accuracy $\sigma(\mathbf{p}_i)$ is calculated by taking the square root of the variance expression equation (9). *stdtol* is the currently allowable tolerance. Starting with a value of *maxstdtol*, it decreases linearly to zero. Hence the GP is only when the accuracy of the prediction is inadequate. The tolerance for prediction error decreases gradual as the search progresses.

The target values of the training data for the GP were normalized to have a mean of zero and a standard deviation of 1. Although, this is not strictly necessary, it was found to improve the numerical stability of the GP and to yield better predictions, as it better satisfies the assumption in equation (2). Thus the standard deviation values predicted by the GP are actually the fraction of the standard deviation of the evolutionary algorithm sampled points that were used to train it. In all the experiments reported here *maxstdtol* was set to 10%.

The algorithm works by first initializing a population of size *Np* and using it to construct the initial GP. Then it proceeds by applying the appropriate evolutionary operators to yield a new progeny population. The predicted standard deviation of each member of the population is then evaluated. If this does not exceed the currently allowable tolerance it is evaluated using the GP, otherwise it is evaluated using the computationally expensive model and that result used to update the GP. If the new population did not result in any GP model update, the allowable tolerance is tightened by a factor of two. This is done to prevent an infinite loop scenario where all evaluations continue to be carried out using the GP. The evolutionary cycle is

then repeated until the maximum number of true evaluations is reached.

Initially the GP's hyperparameters are optimized using the initial population, afterwards simple online model expansion is carried out using equations (14) to (19). When the total number of points added using online model expansion exceeds 40% of the number of points that were last used when the GP model was optimized, a reoptimization step is executed. In this reoptimization step the old hyperparameters are used as initial guess in equations (12) and (13) for carrying out the conjugate gradient search.

5 Experiments on Satellite Boom optimization

A two dimensional satellite boom structure was used to evaluate the proposed EA-GP synthesis. The baseline structure is shown below in Figure 2. It consists of 40 individual Euler-Bernoulli beams connected at 20 joints [15]. Each of the 40 beams has the same properties per unit length. The beam is excited at one end as shown. The goal of the optimization was set at minimizing the frequency averaged response of the end beam in the range 150-250 Hz. The optimizer was allowed to generate new geometries by varying the coordinates of the inner 18 joints of this simplified structure. That is 36 optimization variables representing the *x* and *y* joint positions.

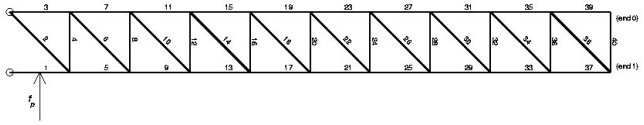

Figure 2 The baseline structure

A typical frequency response is a show below in Figure 3. The optimized structure for this particular graph is shown in Figure 4.

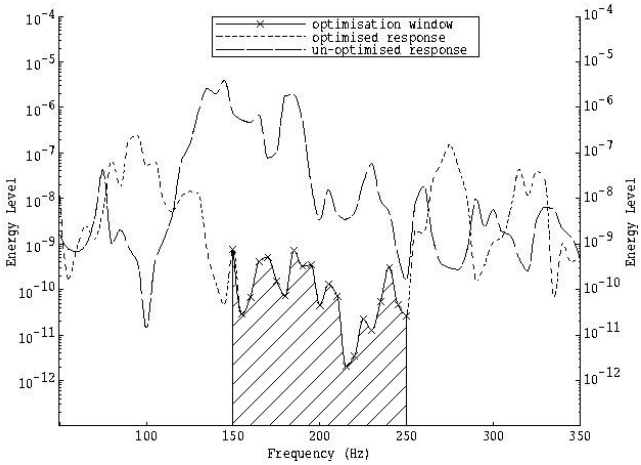

Figure 3 Frequency response of the 2D boom

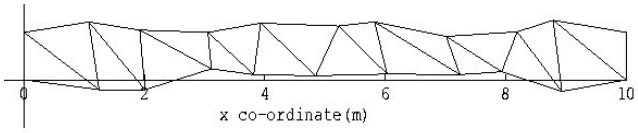

Figure 4 Optimized structure

In this problem, the objective function to be minimized depends on the area under the graph in Figure 3 in the range of 150-250 Hz. The calculation of a single frequency point in this plot requires the solution of some 260 complex simultaneous equations. The more points that are used to perform the integration, the more accurate is the evaluation.

Here, eleven points from the frequency response curve were used to calculate the average energy transmitted in the structure in the range of 150-250 Hz. The frequency response calculations were carried out using matrix receptance methods based of the Green functions of the individual beam elements [16].

The log energy values were used to construct the GP. This pre-processing step was found to be essential, as the GP couldn't cope with making predictions using the raw data, where function target values differ by orders of magnitude (it should be noted that is it a common practice in structural dynamics to use the log of the energy quantities).

A GA typical of those described in Michalewicz [17] was used for the experiments reported here. The GA makes use of arithmetic crossover, gaussian mutation, and the tournament selection operator. The crossover and mutation probabilities were kept constant at 0.9 and 0.15 for all the results reported here. The population size was set to 50, and the

tournament size was set to 2. *maxeval* was set to 500.

A baseline comparison was conducted by running the same GA solely on the accurate function for 500 evaluations. Five runs were carried out for comparison. The average baseline objective function value came to 1.28×10^{-10}. This was used to normalize the results shown in the table below. The results are averaged over five runs.

	Average
Baseline	1
Evolutionary-GP approach	0.86
Evolutionary-GP with initial GP	0.38

Table 1 Normalized energy levels.

Using the approach outlined clearly resulted in an improvement. It was found that during optimization the GP was used only about 20 % of the time. Better results were obtained when an initial GP was used instead of using the evolutionary algorithm's initial population. This initial GP was constructed using 500 randomly sampled points. Once built it was used for all subsequent runs. As the table shows, significant improvements were obtained by using an initial GP. In this latter strategy, the GP was used about 46% of the time. The percentage usage of the GP may be further improved if a higher *maxstdtol* value was used, however this might adversely affect the quality of the optima found by the optimizer as it might be mislead at the early stages of the search.

6 Conclusion and future work

We have presented an approach for combining GP and EAs to make more efficient use of computationally expensive function evaluations. The choice of GP as an approximation tool was justified. The suggested algorithm overcomes some the limitation of earlier approaches, in that the number of adjustable parameters is reduced significantly.

It was applied to the satellite boom optimization problem and positive results were obtained. It is noteworthy to mention that optimizing the hyperparameters for the beam problem was far from trivial. There was a total of 38 hyperparameters and the likelihood function of equation (12) was highly multimodal. Many starting points in the hyperparameter space had to be used to obtain a good GP. This suggests that for problems of large dimensionality, it might be better to use evolutionary search to optimize the hyperparameters, instead of the conjugate gradient optimizer traditional used for GP.

Starting with an initially constructed GP has also shown to be advantageous to using the search data only. So, the more information provided to GP initially the better.

Building and updating the GP is essentially a cumulative process and the computational complexity of its construction is of the order N^3. Hence for a large enough N the cost of constructing a GP will be quite significant. The use of multiple local metamodels is expected to alleviate this burden. This can be achieved by adopting the strategy suggested in [18], where a SOM is used to partition the data space for constructing interpolating models.

Acknowledgments

This work was supported under EPSRC grant no GR/L04733.

Bibliography

[1] A. Ratle, "Optimal sampling strategies for learning a fitness model," in *Proceedings of the 1999 Congress on Evolutionary Computation*, vol. 3. Piscataway, NJ: IEEE Press, 1999, pp. 2078-2085.

[2] A. Ratle, "Accelerating the convergence of Evolutionary Algorithms by Fitness Landscape Approximation," in *Parallel Problem Solving from Nature- PPSN V*, *Springer-Verlag, Lecture Notes in Computer Science*, T. Bäck, A. E. Eiben, M. Schoenauer, and H. P. Schwefel, Eds. Berlin: Springer, 1998, pp. 87-96.

[3] M. A. El-Beltagy, P. B. Nair, and A. J. Keane, "Metamodeling Techniques For Evolutionary Optimization of Computationally Expensive Problems: Promises and Limitations," in *Proceedings of the Genetic and Evolutionary Computation Conference*, vol. 1, W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, Eds. San Francisco, California: Morgan Kaufmann, 1999, pp. 196-203.

[4] T. W. Simpson, J. D. Peplinski, P. N. Koch, and J. K. Allen, "On the Use of Statistics in Design and the Implications for Deterministic Computer Experiments," in *ASME Design Theory and Methodology*, J. Shah, Ed. New York: ASME, 1997, pp. ASME97-DETC97/DTM-3881.

[5] J. Sobieszczanski-Sobieski, "Optimization by Decomposition," in *Structural Optimization: Status and Promis*, G. I. N. Rozvany, Ed. Dordrecht: Kluwer Academic, 1993, pp. 193-233.
[6] J. Sobieszczanski-Sobieski and R. T. Haftka, "Multidisciplinary Aerospace Design Optimization: Survey of Recent Developments," *34th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, AIAA Paper No. 96-0711*, pp. 32, 1996.
[7] P. Hajela and E. Lee, "Topological optimization of rotorcraft subfloor structures for crashworthiness considerations," *Computers and Structures*, vol. 64, pp. 65-76, 1997.
[8] J. F. M. Barthelemy and R. T. Haftka, "Approximation concepts for optimum structural design-a review," *Structural Optimization*, vol. 5, pp. 129-144, 1993.
[9] F. Van Keulen, V. V. Toropov, and A. A. Polynkine, "Shape Optimization Strategies using the Multi-Point Approximation Method and Adaptive Mesh Refinement," *Proceedings of the First World Congress on Structural and Multidisciplinary Optimization. Pergamon*, pp. 67-74, 1995.
[10] V. V. Toropov, "Simulation approach to structural optimization," *Structural Optimization*, vol. 1, pp. 37-46, 1989.
[11] V. V. Toropov, A. A. Filatov, and A. A. Polynkin, "Multiparameter structural optimization using FEM and multipoint explicit approximations," *Structural Optimization*, vol. 6, pp. 7-14, 1993.
[12] N. M. Alexandrov, R. M. Lewis, C. R. Gumbert, L. L. Green, and P. A. Newman, "Optimization with variable-fidelity models applied to wing design," The american institute of aeronautic ans astronuatics (AIAA), 38th Aeropace Sciences Meeting & Exhibit 2000-0841, 2000.
[13] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*. Cambridge: Cambridge University Press, 1986.
[14] R. M. Neal, Bayesian Learning for Neural Networks: Springer, 1996.
[15] A. J. Keane and A. P. Bright, "Passive vibration control via unusual geometries: Experiments on model aerospace structures," *Journal of Sound and Vibration*, vol. 190, pp. 713-719, 1996.
[16] K. Shankar and A. J. Keane, "Energy flow prediction in a structure of rigidly joined beams using receptance theory," *Jouranl of Souand and Vibratation*, vol. 185(5), pp. 867-890, 1995.
[17] Z. Michalewicz, *Genetic Algorithms+Data Structures =Evolution Programs*, 3rd ed. New York: Springer, 1996.
[18] J. Vesanto, "Using the SOM and Local Models in Time-Series Prediction," in *Proceedings 1997 Workshop on Self-Organizing Maps (WSOM'97)*, 1997.