Comparison of Methods for Developing and Using Dynamic Reduced Models for Design Optimization

by

Xiao Ni

(Under the direction of Khaled Rasheed)

Abstract

This thesis compares different methods used in developing and using dynamic reduced models for design optimization. Several methods have been used for forming functional approximation models with which the GA search is performed more efficiently. The informed operators approach for search speedup is discussed. Another hypothesis-instantiation approach in using reduced models has been tested and compared with the informed operator approach. Chapter one first gives some background knowledge. Chapter two describes three different methods for forming reduced models to speed up genetic-algorithm-based design optimization. Chapter three describes two methods for using reduced models to speed up genetic-algorithm-based design optimization. Chapter four gives the experimental results and discussion. Finally, chapter five serves as a summary of the whole thesis.

Index words: GADO, Quickprop, RBF, Quadratic least squares,
Genetic engineering, Informed operators, Reduced models

Comparison of Methods for Developing and Using Dynamic Reduced Models for Design Optimization

by

Xiao Ni

B.S., Xian Jiaotong University, China, 2000

A Thesis Submitted to the Graduate Faculty
of The University of Georgia in Partial Fulfillment
of the
Requirements for the Degree

Master of Science

Athens, Georgia

2002

Comparison of Methods for Developing and Using Dynamic Reduced Models for Design Optimization

by

Xiao Ni

Approved:

Major Professor: Khaled Rasheed

Committee: Ron McClendon
Charles Cross

Electronic Version Approved:

Gordhan L. Patel
Dean of the Graduate School
The University of Georgia
July 2002

Acknowledgments

I would like to thank my advisor, Khaled Rasheed, for his consistent support and guidance in my master's study. I thank Ron McClendon for his invaluable advice in my thesis work. I thank Charles Cross for being on my thesis committee. I also thank my fellow student and friend, Swaroop Vattam, for his collaboration and the implementation of the RBF part in this project.

Next I thank my parents and my brother, for their continued help and encouragement. I thank my friends and fellow students, Ning Suo, Jason Schlacheter, Tong Wang, and Chun Liang for their kind help. It is lucky for me to meet them and work together for almost two years.

This research was funded by a sub-contract from the Rutgers-based Self Adaptive Software project supported by the Advanced Research Projects Agency of the Department of Defense and by NASA under grant NAG2-1234.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The Engineering Design Optimization Problem

Optimization is a process of finding and comparing feasible solutions until an optimal solution is identified. In the real world, the engineering design optimization problem is to use some optimization method, coupled with a simulator or some engineering analysis code, to find the best design according to some measure of merit and subject to some constraints.

Basically there are two kinds of design problems: structural and parametric design. In this thesis, we just focus on continuous parametric design, especially the realistic problem of aircraft design. This means the main goal is to optimize the key parameters of an aircraft, such as wing area, length and so on. The detailed design process is very complicated, and we need to reduce this to a general constrained nonlinear programming problem with a reasonable number of variables, in which the measure of merit (objective) and constraints serve as the evaluation of a certain design. This is the conceptual design stage.

In a general form, the design optimization problem is represented as follows:

$$
\begin{aligned}
&\textit{minimize } f(\overline{x}) \\
&\textit{subject to:} \\
&g_i(\overline{x}) \leq 0 \ i = 1, ..., l \\
&h_j(\overline{x}) = 0 \ j = 1, ..., m
\end{aligned}
$$

$\overline{x}^{(L)} \leq \overline{x} \leq \overline{x}^{(U)}$ $j = 1, ..., m$

where

- $\overline{x}$ represents the vector consisting of real number parameters of the object that is being designed. $\overline{x}^{(L)}$ and $\overline{x}^{(U)}$ correspond to the lower and upper bound of the parametric vector respectively.
- $f(\overline{x})$ is the objective function, which is to be optimized.
- $g_i(\overline{x})$ and $h_j(\overline{x})$ are inequality constraints and the equality constraints respectively.

Once the mathematical model of the design problem has been built, one can use a variety of optimization methods to find an optimal solution.

1.2 GADO: Genetic Algorithm for Design Optimization

The classical numerical optimization methods are mostly gradient based. These methods turn out to be inadequate to deal with the real-world design problems because of the difficult features of engineering design domains. Most of these methods are prone to get stuck in local optima and fail to reach the global optimum.

Since the mid eighties of the last century, Evolutionary Algorithms have been studied in depth and applied in difficult design problems. Compared with the conventional techniques, Evolutionary Algorithms can find multiple optimal solutions in one single optimization run due to their population-approach. During the last decade, Genetic Algorithms have been significantly improved and adapted to various engineering domains and became powerful and broadly applicable stochastic search and optimization techniques.

This thesis concerns the application of Genetic Algorithms (GAs) in realistic engineering design domains. In such domains a design is represented by a number

of continuous design parameters, so that potential solutions are vectors (points) in a multidimensional vector space. Determining the quality ("fitness") of each point usually involves the use of a simulator or some analysis code that computes relevant physical properties of the artifact represented by the vector, and summarizes them into a single measure of merit and, often, information about the status of constraints. For example, the problem may be to design a supersonic aircraft capable of carrying 70 passengers from Chicago to Paris in 3 hours. The goal may be to minimize the takeoff mass of the aircraft. The constraints may include something like "the wings must be strong enough to hold the plane in all expected flight conditions".

We conducted our investigations in the context of GADO [11, 14], a GA that was designed with the goal of being suitable for use in engineering design. It uses new operators and search control strategies suitable for the domains that typically arise in such applications. GADO has been applied in a variety of optimization tasks that span many fields. It demonstrated a great deal of robustness and efficiency relative to competing methods.

In GADO, each individual in the GA population represents a parametric description of an artifact, such as an aircraft or a missile. All parameters take on values in known continuous ranges. The fitness of each individual is based on the sum of a proper measure of merit computed by a simulator or some analysis code (such as the takeoff mass of an aircraft), and a penalty function if relevant (such as to impose limits on the permissible size of an aircraft). The penalty function consists of an adaptive penalty coefficient multiplied by the sum of all constraint violations if any. A steady state GA model is used, in which operators are applied to two parents selected from the elements of the population via a rank based selection scheme, one offspring point is produced, then an existing point in the population is replaced by the newly generated point via a crowding replacement strategy. Floating point representation is used. Several crossover and mutation operators are used, most of which

were designed specifically for the target domain type. GADO also uses a search-control method [14] that saves time by avoiding the evaluation of points that are unlikely to correspond to good designs.

1.3 Dynamic Reduced Models

1.3.1 General Description

Although GAs have been successfully applied to many engineering design domains, one of the major problems is the heavy-duty fitness computation by the simulator. It is often the case that the simulator will take a non-negligible amount of time to evaluate a point. A GA search may need a large number of iterations before it arrives at the global optimum, which has significantly influenced the feasibility of GAs for these design problems.

Specifically, some of the problems faced in the application of GAs (or any optimization technique for that matter) to such problems are:

- Not all points in the space are legitimate designs — some points in the search space ("unevaluable points") cause the simulator to crash, and others ("infeasible points"), although evaluable by the simulator, do not correspond to physically realizable designs.

- The simulator will often take a non-negligible amount of time to evaluate a point. The simulation time ranges from a fraction of a second to, in some cases, many days.

- The fitness function may be highly non-linear. It may also have all sorts of numerical pathologies such as discontinuities in function and derivatives, multiple local optima, ..etc.

Fortunately, in many of these domains so-called "reduced models", which provide less-accurate but more efficient estimates of the merit of an artifact, are either readily available or can be learned online (i.e. in the course of the optimization) or off-line (i.e. by sampling and building a response surface before optimization). This thesis compares methods for the modification of GAs specifically intended to improve performance in realistic engineering design domains in which no reduced models are available a priori. These methods form approximations of the fitnesses of the points encountered during the course of the GA optimization. The approximations are then used to speed up the GA by making its operators more informed.

The use of reduced models to save time in evolutionary optimization dates all the way back to the sixties. Dunham et al. [2] worked with a two level problem in which they used an approximate model most of the time and only used the accurate/expensive model in the final stages of refinement. Numerous research efforts compute a response surface approximation and use it instead of the very expensive evaluation function with no looking back [18]. Other approaches rely on special relations between the approximate and accurate model to develop interesting multi-level search strategies. A notable class of such methods [3] focus on building variants of injection island genetic algorithms (iiGAs) for problems involving finite element analysis models. The approach was to have many islands using low accuracy/cheap evaluation models with small numbers of finite elements that progressively propagate individuals to fewer islands using more accurate/expensive evaluations. A recent approach [4] uses a functional approximation method to form reduced models. To the best of our knowledge, none of these approaches addressed the problem of unevaluable points.

1.3.2 Current approach to form reduced models

We conducted our investigation in the context of the framework described in detail in [13]. Our method is based on maintaining a large sample of the points encountered in the course of the optimization. Ideally, the sample should include all the points, but if the simulator is relatively fast or the optimization takes a relatively high number of iterations we maintain a smaller sample in order to keep the overhead of reduced model formation and use reasonable.

Least squares approximations of the measure of merit and constraints are formed periodically, for both the global set and each large cluster. We used the normal equations implementation from *Numercial Recipes in C* [10] to form quadratic least squares approximations.

1.3.3 The Artificial Neural Networks approach

Besides the least squares method, we decided to further investigate other ways for forming reduced models, hoping to get more accurate and effective approximation. The Artificial Neural Networks (ANNs) simulate the networks of neurons in human brain. A three-layer backpropagation ANN with sufficient hidden nodes is supposed to be able to learn a function of arbitrary complexity or non-linearity. As one of powerful approximation tools, the ANN works well by generalizing from a given patterns to make very precise predictions.

Our work is to replace the least squares method with an Artificial Neural Network model, based on which we build the reduced model. We used the points that we have encountered in the optimization process as ANN input patterns. The networks are periodically updated and the structural features are saved in some data structures for future prediction use.

Two kinds of ANN structures have been used in our work, one is *Quickprop algorithm* [8], and the other is *Radial basis function network (RBFN)*. Quickprop algorithm [5] is essentially a variant of the standard backpropagation neural network, which makes use of the second order derivative information to learn faster. On the other hand, the radial basis function networks are class of single hidden layer feed forward networks where the activation functions for hidden units are radially symmetric basis functions such as Gaussian function. The RBFN part was done by Swaroop Vattam. The details about ANN-based reduced model are described in chapter two.

1.4 Using the Dynamic Reduced Models

We tested two kinds of approaches to use reduced models, the informed operators approach and the genetic engineering approach. Our purpose is to speed up the design optimization with the help of reduced models.

With the dynamic reduced model, we can make the genetic operators more informed, as well as significantly reduce the computation workload of using the simulators. The idea of informed genetic operators is to generate several candidates instead of only one, and rank them using reduced models. Then we only pick the potentially best point and let it go to the next stage. If the computational complexity of maintaining and using the reduced model is negligible compared to the time of running the real simulator, this scheme may significantly speed up the GA search. On the other hand, another method for using the reduced model has been investigated and compared to the existing way. We use the genetic engineering operator to create some new individuals instead of the traditional Darwinian genetic operators. The basic idea is to use some functional minimization methods to find the minimal point, and this point will be treated the same way as those newborns generated by

Darwinian genetic operators. The reduced model serves as the functional evaluator. Chapter three describes the comparison of these two methods.

Chapter 2

Methods of Developing Dynamic Reduced Models

2.1 General Framework

We conducted our investigation in the context of the framework described in detail in [13]. We provide only a brief description here. Our method is based on maintaining a large sample of the points encountered in the course of the optimization. Ideally, the sample should include all the points, but if the simulator is relatively fast or the optimization takes a relatively high number of iterations we maintain a smaller sample in order to keep the overhead of reduced model formation and use reasonable.

- **Incremental approximate clustering** We keep the sample divided into clusters. Starting with one cluster, we introduce one more cluster every specific number of iterations. The reason we introduce the clusters incrementally rather than from the beginning is that this way results in more uniform sized clusters. Every new point entering the sample, either becomes a new cluster (if it is time to introduce a cluster) or joins one of the existing clusters. A point belongs to the cluster whose center is closest in Euclidean distance to the point at the time in which the point joined the sample.

- **Approximate evaluation of new points** The first step in evaluating the approximate fitness of a new point is to find to which cluster it belongs. If the point belongs to a cluster with cluster approximation functions, these are to be used, otherwise the global approximation functions are to be used. The

evaluation method depends on the stage of the optimization. In the first half of the optimization the fitness is formed by using the approximate measure of merit and the approximate sum of constraints (which is forced to zero if negative). No attempt is made to guess at whether the point will be feasible, infeasible or unevaluable. In the second half of the optimization we use a two phase approach. First we use the nearest neighbors of the new point to guess whether the point is likely to be feasible, infeasible-evaluable or unevaluable. Based on this guess, and the point's cluster, we then use the proper approximation functions (for example, no approximation functions are used if the point is guessed to be unevaluable).

2.2 Quadratic Least Squares Approximations

The first approach we used for forming the approximations was Quadratic Least Squares (LS). We distinguish between the approximation functions for the measure of merit and those for the sum of constraints.[1] The reason is that the constraints are only defined for infeasible designs. For feasible designs we have to put all the constraints at zero level as the simulators only return that they are satisfied. We form two types of approximations for measure of merit and for the sum of constraint violations:

- **Global approximation functions**

  We maintain two global approximation functions which are based on all the **evaluable** points in the sample.

  We use quadratic approximation functions of the form:

$$\hat{F}(\bar{X}) = a_0 + \sum_{i=1}^{n} a_i x_i + \sum_{i=1,j=i}^{n,n} a_{ij} x_i x_j$$

[1]Since GADO only deals with the sum of all constraint violations rather than the individual constraints, we only form approximations for the sum of all constraint violations.

where n is the dimension of the search space.

We use a least square fitting routine from [10]. It works by using the normal equations method.

- **Cluster approximation functions**

  We use the same techniques for forming cluster approximation functions, except that we only form them for clusters which have a sufficient number of evaluable points.

2.3 Radial Basis Function Neural Networks

We employ two Gaussian radial basis function (RBF) neural networks, one to compute the approximation for the measure of merit and the other to compute the constraint violations for each large cluster as well as the whole sample. The structure of each of the RBF networks is as reported in some parts of the work by Howell and Buxton [7]. It consists of an input layer, a hidden layer with nonlinear activation functions, and an output layer with linear activation function. The size of the RBF network is determined by 1) the dimension of the domain, 2) the number of training examples, which gives the number of hidden units, and 3) one output neuron for estimating the values of measure of merit or the constraint violations.

The basic version of the RBF network had to be modified in order to integrate it into GADO. Learning rate, a variable parameter of the RBF network had to be tuned according to the domain in order to achieve optimal performance. To overcome this, we made the learning rate adaptive. In every domain, we start with a fixed value for the learning rate, and modify it in the positive or the negative direction in small increments, testing for the mean square error of the training and the testing set at regular intervals. This process ensures that the learning rate used, eventually, will

produce the best results for that domain. We also added a mechanism to avoid over-training the network based on using an automated training-testing routine in which the network training is halted periodically at predetermined intervals (calibration), and the network is then run in recall mode on the test set to evaluate the network's performance on MSE (mean squared error). Specifically, the MSE of the training set should decrease, and so should the test set MSE, otherwise, we stop training the networks. The best saved network configurations - up to this point - are then used for further prediction. With these changes in place, the RBF networks were integrated into GADO and the results obtained are compared with the other approaches in the subsequent sections. For further information regarding the implementation aspects of the RBF networks, please refer to [7].

2.4 Quickprop Neural Networks

Quickprop is a modification of the back-propagation learning algorithm (Backprop) that uses a second-order weight-update function, based on measurement of the error gradient at two successive points, to accelerate the convergence over simple first-order gradient descent [5]. It is loosely based on Newton's method, but with less computational complexity. It computes estimates of the error minimum using the information of the last weight change, the last slope and the current slope. The following equation describes the calculation of a weight change, with respect to the derivatives and the last weight change.

$$\Delta w_t = \frac{s_t}{s_t - s_{t-1}} \Delta w_{t-1}$$

where

- Δw_t is the current weight change
- Δw_{t-1} is the last weight change

- s_t is the current slope (derivative)
- s_{t-1} is the last slope (derivative)

Quickprop learns much faster than the standard back-propagation but also has more parameters that have to be fine-tuned[5]. In this work, we used the C version of Quickprop, translated by Terry Regier from Fahlman's original Lisp version. The implementation is described in www.cs.ncl.ac.uk/modules/1998-99/csc325/projects/quickprop/.

There are two measures to be approximated, the measure of merit and the constraint violations, and therefore, the network structure could be either one-network-two-output, or two-network-one-output. We used the one-network-two-output structure for the consideration of memory cost and computational complexity, because two-network-one-output structure will almost double the space and time complexity. The number of hidden nodes was set to 26, which is about twice the biggest dimension of our test domains. While this is a relatively large number, we relied on the overfitting detection mechanism and the results are reasonable. The maximum number of training epochs was 800. We have used the maximum growth factor of 1.75, as was recommended in Fahlman's empirical report [5]. The weight decay is set to -0.0001, as used in [5]. After trial and error experiments, we found that a learning rate of 0.01 is good.

As with the RBF network, we introduced a mechanism for avoiding over-training in this network. The whole pattern set (either the global sample set or one of the large enough cluster sets) are partitioned into two smaller sets: 80% goes to the training set and the other 20% forms the test set. We monitor the MSE (Mean Squared Errors) of both the training and test sets periodically (every 50 iterations, we call it a calibration). First we make sure the MSE of the training set decreases since the last calibration, and then we compare the MSE of the test set with the value we get

in the last calibration. If the test set MSE goes up, it indicates that the network is over-trained and thus we halt the training. With overfitting control, we stop training the neural networks before the overfitting occurs. After testing this mechanism, we found there is overfitting that takes place before the maximum number of training epochs is reached.

We employed a scaling and normalizing data pre-processing procedure before we trained the neural networks. The real pattern output values must be scaled to the range of the logistic function, because otherwise we will not get the correct error measure. A linear-scaling method is used to scale both the inputs and outputs. We also used Z-score normalization to normalize the inputs. We have found the data pre-processing procedure necessary and helpful for the neural network training. On the other hand, we should scale the output values back to the real range, when we use the reduced models to approximate the cheap fitnesses.

The Quickprop algorithm was implemented and integrated into GADO so as to use both the global and the cluster approximation models. We form the approximations for measure of merit and constraint violations by maintaining both a global ANN and an ANN for each big enough cluster (i.e., cluster with a sufficient number of evaluable points) in a manner similar to that used for LS and RBF approximations.

2.5 Comparison of Reduced Models Formation Methods

To compare the performance of the different approximation methods, we used reduced models that are acquired on-line to create informed genetic operators. These operators are described in detail in [15] where their use was demonstrated with pre-existing reduced models as well as on-line approximation models. The main idea is to make the genetic operators such as mutation and crossover more informed using reduced models. In every place where a random choice is made, for

example when a point is mutated, instead of generating just one random mutation we generate several, rank them using our reduced model, then take the best to be the result of the mutation. We experimented with informed mutation, crossover and initial population formation.

Chapter 3

Methods for using Reduced Models to Speed Up Design Optimization

We compare two methods for improving the GA's performance. One is the idea of informed operators presented in [15] and the other is a variation of the genetic engineering idea presented in [1]. The remainder of this chapter describes these two approaches in more detail.

3.1 Informed operators

The main idea of Informed Operators (IO) is to replace pure randomness with decisions that are guided by the reduced model. We replace the conventional GA operators such as initialization, mutation and crossover with four types of informed operators:

- **Informed initialization:** For generating an individual in the initial population we generate a number of uniformly random individuals in the design space and take the best according to the reduced model. The number of random individuals is a parameter of the method with a default value of 20.

- **Informed mutation:** To do mutation several random mutations are generated of the base point. Each random mutation is generated according to the regular method used in GADO [11] by randomly choosing from among several different mutation operators and then randomly selecting the proper parameters for the mutation method. The mutation that appears best according to

the reduced model is returned as the result of the mutation. The number of random mutations is a parameter of the method with a default value of five.

- **Informed crossover:** To do crossover two parents are selected at random according to the usual selection strategy in GADO. These two parents will not change in the course of the informed crossover operation. Several crossovers are conducted by randomly selecting a crossover method, randomly selecting its internal parameters and applying it to the two parents to generate a potential child. The internal parameters depend on the crossover method selected. For example to do point crossover the cut-and-paste point has to be selected. Informed mutation is applied to every potential child, and the best among the best mutations is the outcome of the informed crossover. The number of random crossovers is a parameter of the method with a default value of four. Thus each crossover-mutation combination uses 20 reduced model evaluations.

- **Informed guided crossover:** Guided crossover [12] is a novel operator used in GADO to replace some of the regular crossover-mutation operations to improve convergence towards the end of the optimization. Guided crossover does not involve mutation so we treat it differently. The way informed guided crossover works is a follows:

  - Several candidates are selected at random using the usual selection strategy of GADO to be the first parent for guided crossover. The number of such candidates is a parameter of the method with a default value of four.

  - For each potential first parent the second parent is selected in a fashion documented elsewhere [12]. Then several random points are generated from the guided crossover of the two parents and ranked using the reduced

model. The number of such random points is a parameter of the method with a default value of five.

– The best of the best of the random points generated is taken to be the result of the guided crossover.

Thus the default total number of reduced model calls per informed guided crossover is 20.

3.2 Genetic Engineering

Our approach to Genetic Engineering (GE) attempts to improve the efficiency of the GA optimization by replacing some of the regular Darwinian iterations, which generate new individuals using crossover and/or mutation, with iteration in which individuals are generated by running a mini-optimization using the approximations and returning the best point found therein.

In our implementation of the GE approach, the non-gradient based Downhill-Simplex method is used - on a periodic basis - to generate new individuals instead of the traditional genetic operators. The Downhill-Simplex technique [10], is used because of its lesser dependence on the accuracy of the approximation function in comparison to the various gradient-based techniques. This method requires only function evaluations, not derivatives. Once in every four GA iterations we use the Genetic Engineering approach to create a new individual instead of the Darwinian genetic operators. Specifically, assuming the dimension of the parameter space is ndim, we select ndim+1 individuals from the current population using the regular selection strategy of GADO. We then use the cheap fitness function to get the approximate fitness values of these ndim+1 individuals. These ndim+1 individuals and their fitnesses serve as input to the Downhill-Simplex function. Based on the approximated fitnesses, new hypothesized minimum can be found by calling

the Downhill-Simplex function. This minimum serves as the new born individual. Therefore, new individuals are created and evaluated by the true fitness function, so that the overall GA search proceeds.

It was hard to set the number of calls to the cheap fitness function with the GE as each mini-optimization could take a different number of iterations. We set the maximum number of calls during each mini-optimization to 500. However, we found that the GE approach ended up calling the cheap fitness function more than an order of magnitude more times than the IO approach. We did not repeat the experiments because the final conclusion was in favor the IO approach anyway.

Chapter 4

Experimental Results and Discussion

In this chapter, we give experimental results of the method comparisons in developing and using dynamic reduced models. We plotted the fitness value against iteration number to get a curve, which indicates the merit of the corresponding method (the lower the better, as all problems are minimizations).

4.1 Domain Description

We did experiments in one realistic domain–the aircraft design domain, and several other benchmark engineering design domains, which were first introduced in Eric Sandgren's Ph.D. thesis [17].

4.1.1 Supersonic transport aircraft design domain

Our first domain concerns the conceptual design of supersonic transport aircraft. We summarize it briefly here; it is described in more detail in [6]. Figure 4.1 shows a diagram of a typical airplane automatically designed by our software system. The GA attempts to find a good design for a particular mission by varying twelve of the aircraft conceptual design parameters over a continuous range of values. An optimizer evaluates candidate designs using a multidisciplinary simulator. In our current implementation, the optimizer's goal is to minimize the takeoff mass of the aircraft, a measure of merit commonly used in the aircraft industry at the conceptual design stage.

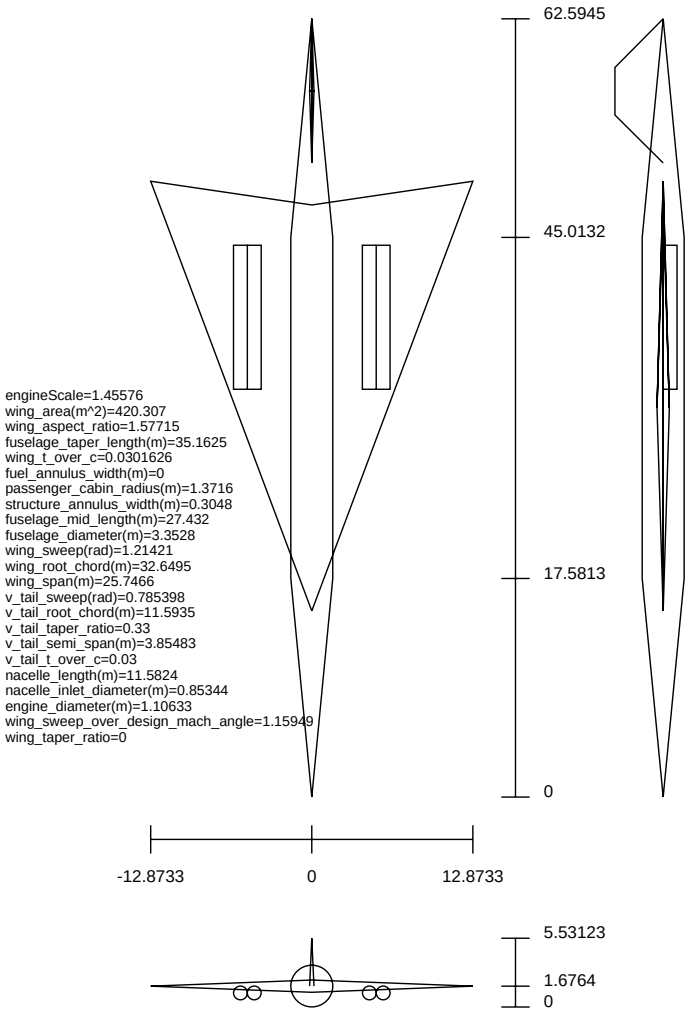

Figure 4.1: Supersonic transport aircraft designed by our system

As shown in Table 4.1, the problem has 12 parameters and 37 inequality constraints. However, only 0.6% of the search space is evaluable.

4.1.2 Benchmark engineering design domains

In order to further compare the different model generation and speedup methods, we compared their performance in several benchmark engineering design domains. These domains were introduced by Eric Sandgren in his Ph.D. thesis [17] in which he

Table 4.1: Aircraft Parameters to Optimize

No.	Parameter
1	Exhaust nozzle convergent length (l_c)
2	Exhaust nozzle divergent length (l_d)
3	Exhaust nozzle external length (l_e)
4	Exhaust nozzle radius (r_7)
5	Engine size
6	Wing area
7	Wing aspect ratio
8	Fuselage taper length
9	Effective structural t/c
10	Wing sweep over design match angle
11	Wing taper ration
12	Fuel annulus width

applied 35 nonlinear optimization algorithms to 30 engineering design optimization problems and compared their performance. Those problems have become used in engineering design optimization domains as benchmarks [9].

Table 4.2 summarizes some properties of the 7 benchmark domains used in our experiments. The second column of the table shows the domain number appearing in Sandgren's thesis. The third column shows the problem dimensions, and the fourth and fifth columns gives the number of inequality and equality constraints respectively. The sixth column shows the best optimization results so far.

4.2 Experimental Results of Comparison of Reduced Models Formation Methods

To compare the performance of the different approximation methods we compare GADO with reduced-model-based informed operators based on each approximation

Table 4.2: Results of Optimization in Benchmark Domains

Domain No.	Sandgren No.	Dim.	Inequality Constraints	Equality Constraints	Best
1	13	5	4	0	26.78
2	2	3	2	0	-3.3
3	3	5	6	0	-3.06
4	6	6	0	4	8.92
5	8	3	2	0	-5.68
6	21	13	13	0	97.5
7	22	16	19	0	174.7

method with each other. We also compare all of these to GADO without reduced-model-based informed operators altogether. We compared the four systems in several domains: the aircraft design domain and benchmarks described above.

4.2.1 Experiments and results in the aircraft design domain

Figure 4.2 shows the performance comparison in domain 1 (aircraft design). Each curve in the figure shows the average of 15 runs of GADO starting from random initial populations. The experiments were done once for each approximation method (LS, QP and RBF) in addition to once without the reduced-model-based informed operators altogether, with all other parameters kept the same. Figure 4.2 demonstrates the performance with each of the three approximation methods as well as performance with no approximation at all (the solid line) in domain 1 . The figure plots the average (over the 15 runs) of the best measure of merit found so far in the optimization as a function of the number of iterations. (From now on we use the term "iteration" to denote an actual evaluation of the objective function, which is usually a call to a simulator or an analysis code. This is consistent with our goal of

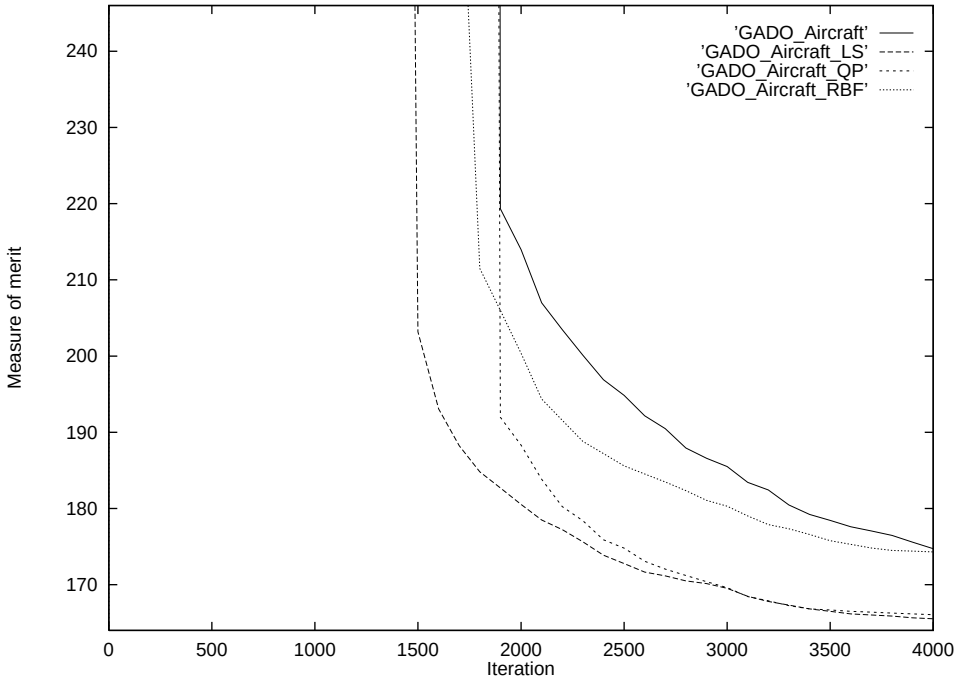

Figure 4.2: Comparison of reduced model formation methods in domain 1 (aircraft design)

understanding how the reduced-model-based informed operators affect the number of calls to the objective function in problems where the informed operators overhead is minuscule compared to the runtime of each objective function evaluation, as was the case here. This also helps us avoid the pitfalls of basing evaluations on run times, which can vary widely — for example across platforms and even across runs due to variations in memory available and hence caching effects.) The figure shows that the LS approximation method gave the best performance in all stages of the search in this domain.

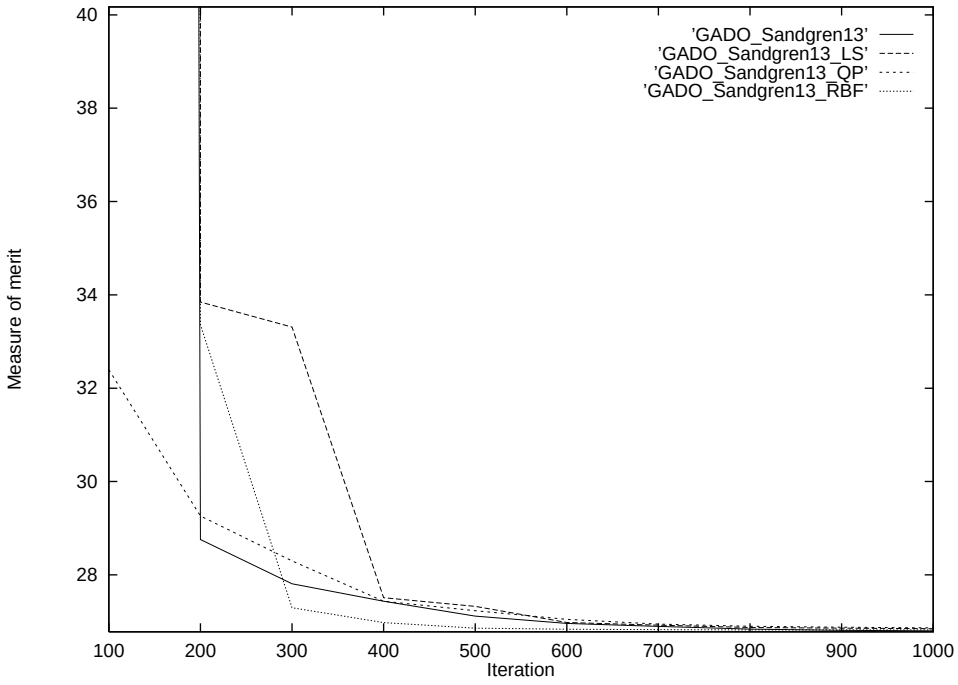

Figure 4.3: Comparison of reduced model formation methods in benchmark domain 1

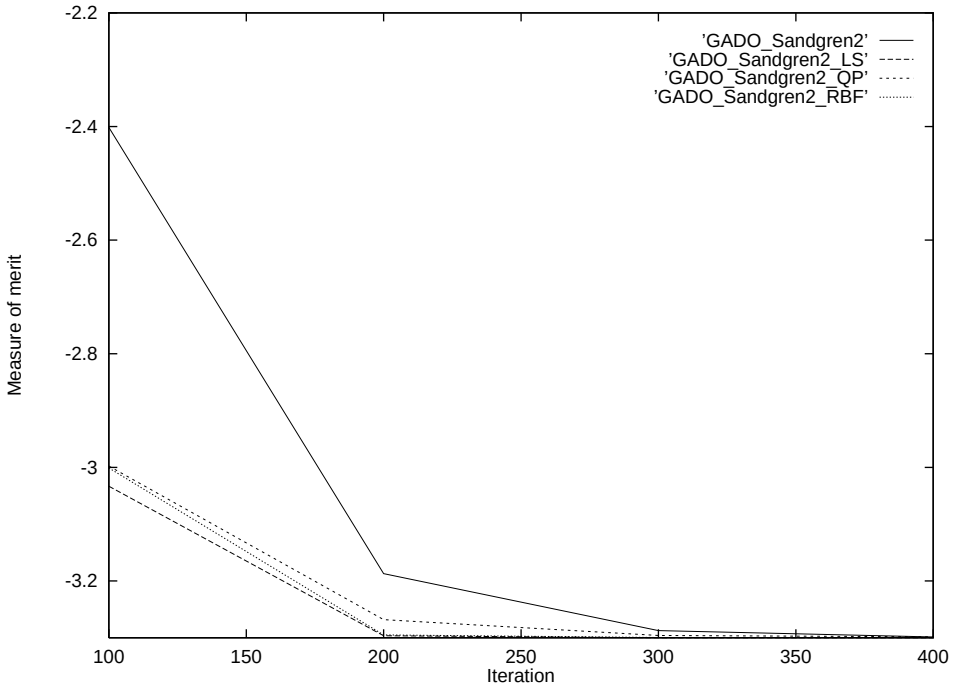

Figure 4.4: Comparison of reduced model formation methods in benchmark domain 2

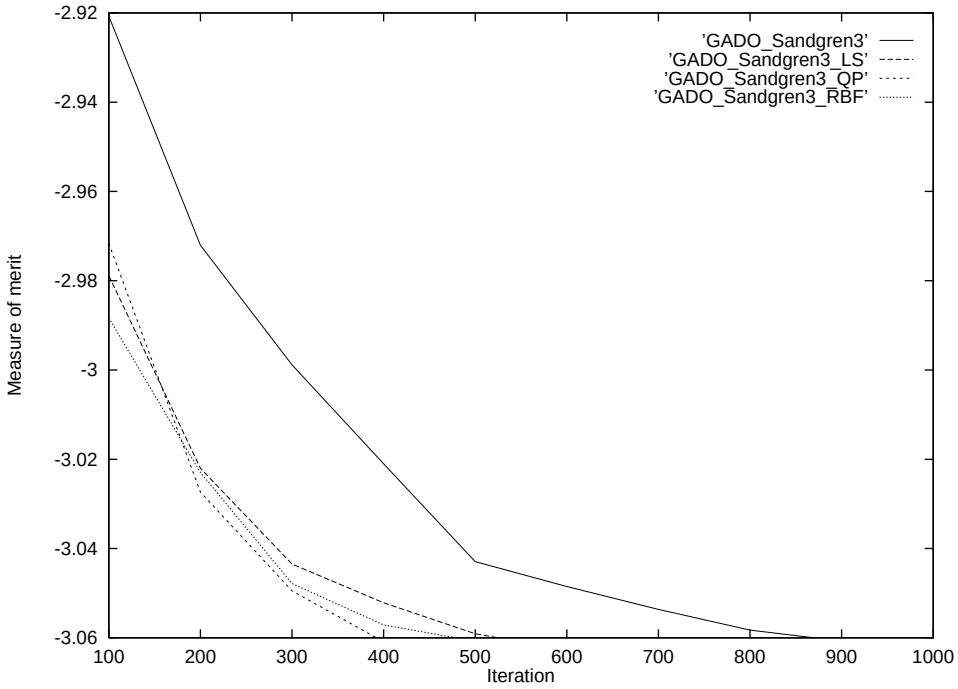

Figure 4.5: Comparison of reduced model formation methods in benchmark domain 3

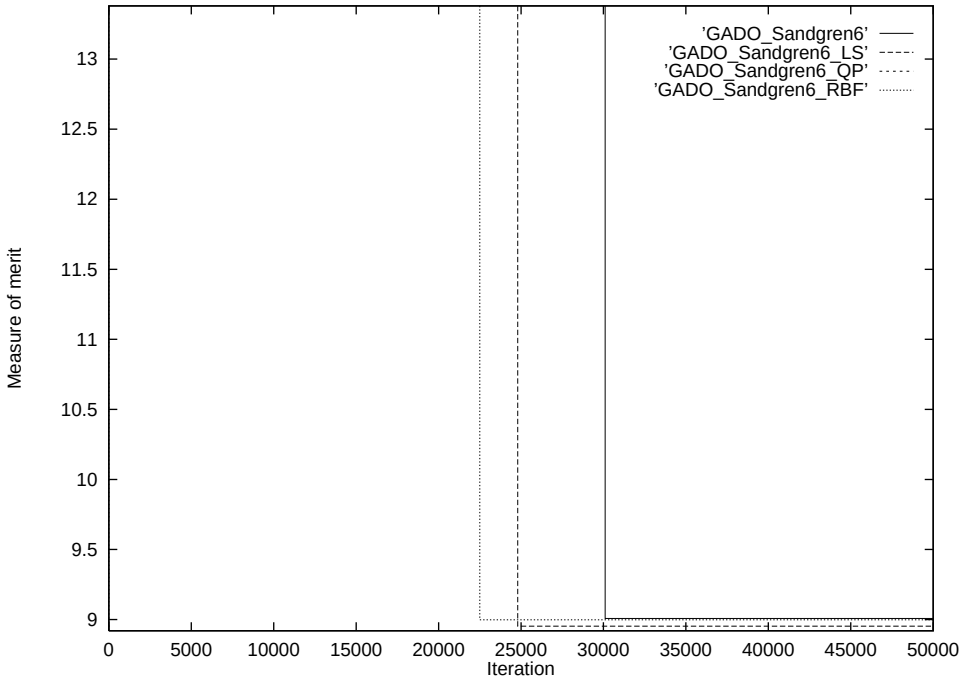

Figure 4.6: Comparison of reduced model formation methods in benchmark domain 4

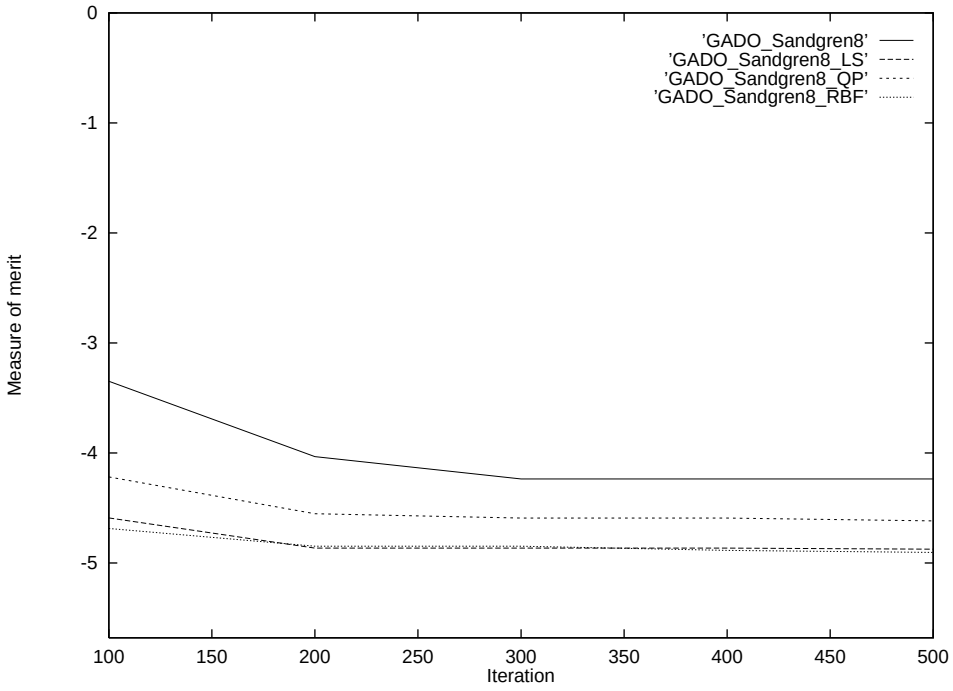

Figure 4.7: Comparison of reduced model formation methods in benchmark domain 5

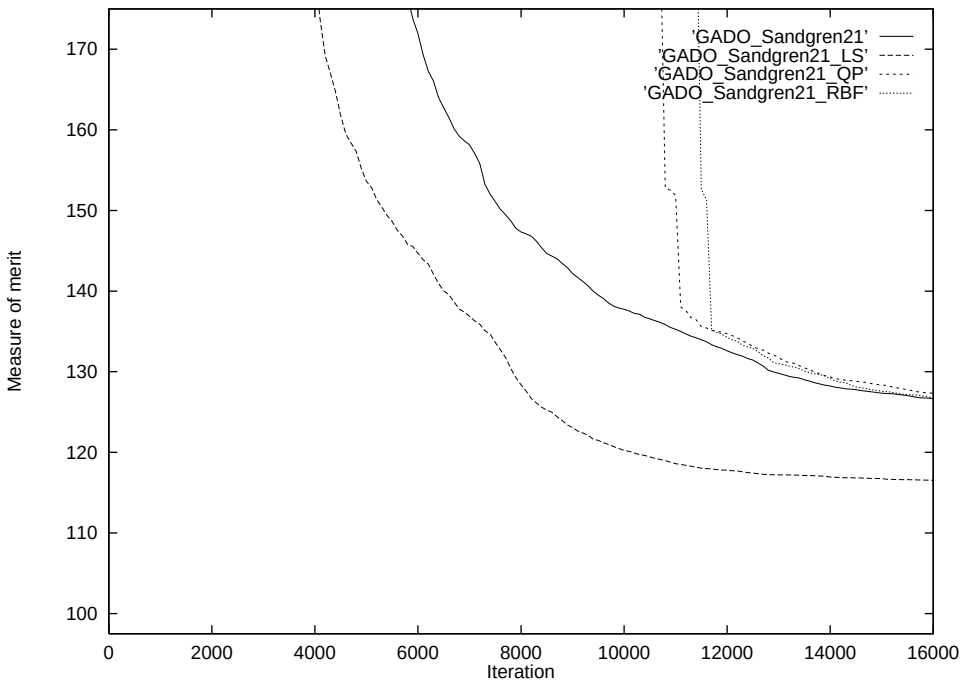

Figure 4.8: Comparison of reduced model formation methods in benchmark domain 6

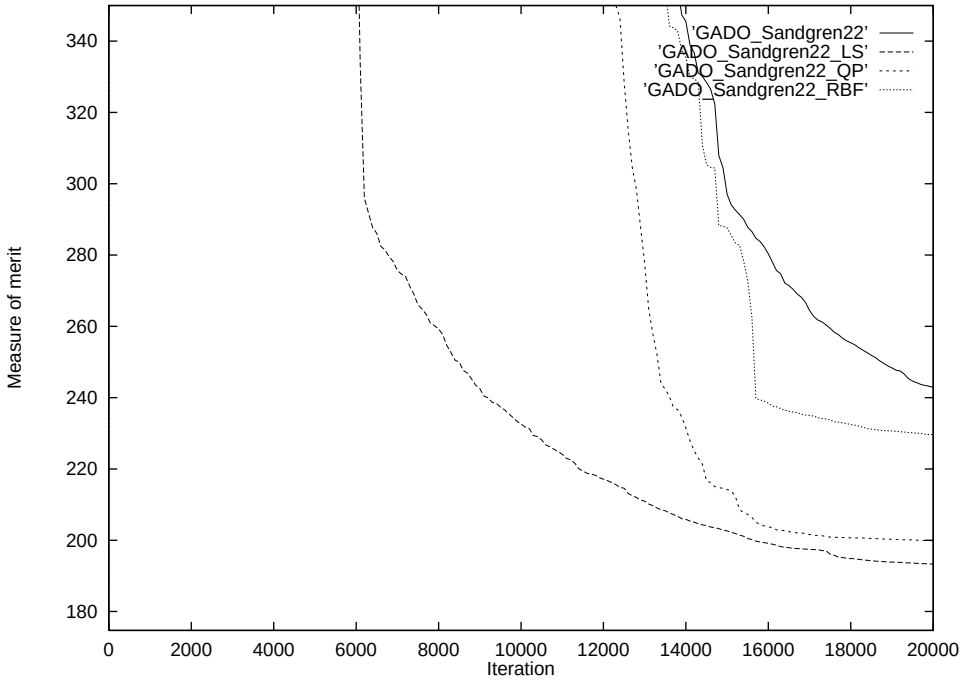

Figure 4.9: Comparison of reduced model formation methods in benchmark domain 7

4.2.2 Experiments and results in benchmark engineering design domains

For each problem GADO was run 15 times using different random starting populations. The experiments were done once for each approximation method (LS, QP and RBF) in addition to once without the reduced-model-based informed operators altogether, with all other parameters kept the same. Figure 4.3 through Figure 4.9 show the performance with each of the three approximation methods as well as performance with no approximation at all (the solid lines) in the benchmark domains. Each figure shows the average of 15 runs of GADO with each of the three approximation methods and once without the informed operators. We found that in the first four benchmarks, which represent relatively easy optimization tasks, the performance

differences were small and it did not make much difference which approximation method was used. The figures also show that the RBF method gave the best final performance in domain 5 while the LS method did much better than the other two in domains 6 and 7 – the two most challenging of all these benchmarks. In fact, the results with RBF and QP approximations in benchmark 7 were worse than with no approximations at all.

4.3 Experimental Results of Comparison of Reduced-model-based Speedup Methods

To compare the performance of the different speedup methods we compared GADO with reduced-model-based informed operators to GADO with genetic engineering. We did the comparisons in the context of LS as well as QP based approximation methods. We also compare all of these to GADO without speedup altogether. We compared the five systems in several domains: the aircraft design domain and benchmarks described earlier.

4.3.1 Experiments and results in the aircraft design domain

Figure 4.10 shows the performance comparison in domain 1 (aircraft design). Each curve in the figure shows the average of 15 runs of GADO starting from random initial populations. The experiments were done once for each speedup method-approximation method combination (i.e. IO with LS,GE with LS, IO with QP and GE with QP) in addition to once without any speedup or approximation methods, with all other parameters kept the same. Figure 4.10 demonstrates the performance with each of the four combinations as well as the performance with no approximation or speedup at all (the solid line) in domain 1 . The figure plots the average (over the 15 runs) of the best measure of merit found so far in the optimization as a function

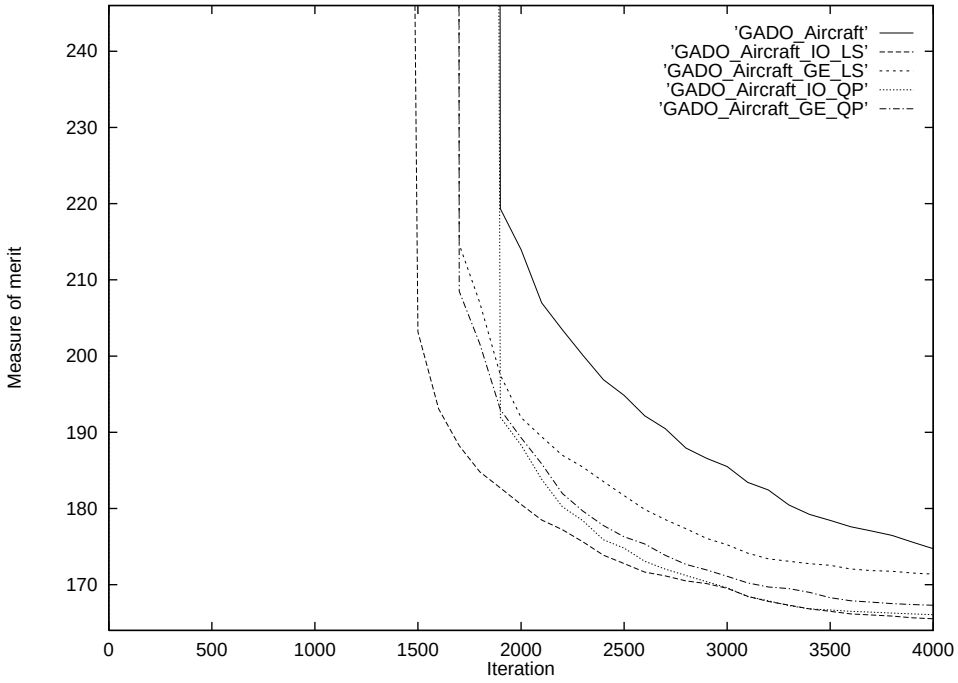

Figure 4.10: Comparison of methods for using reduced models in domain 1 (aircraft design)

of the number of iterations. The figure shows that the informed operators method improved the performance more than the genetic engineering method in most stages of the search regardless of which approximation method was used for forming the reduced models.

4.3.2 Experiments and results in benchmark engineering design domains

For each problem GADO was run 15 times using different random starting populations. As with the aircraft domain, the experiments were done once for each speedup method and approximation method combination, in addition to once without any speedup, with all other parameters kept the same. Figure 4.11 through Figure 4.17

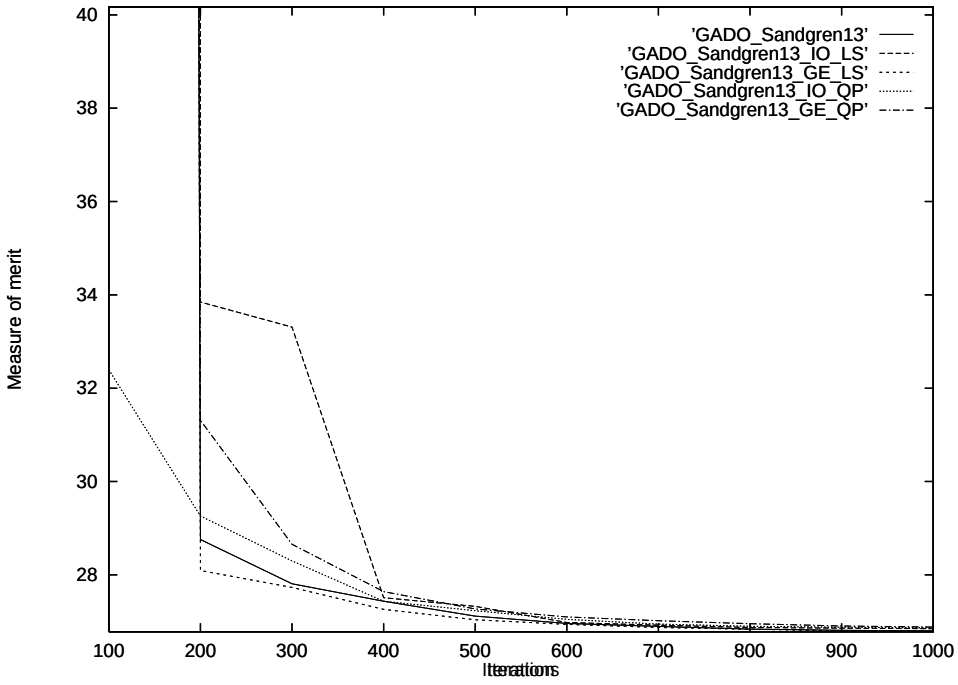

Figure 4.11: Comparison of methods for using reduced models in benchmark domain 1

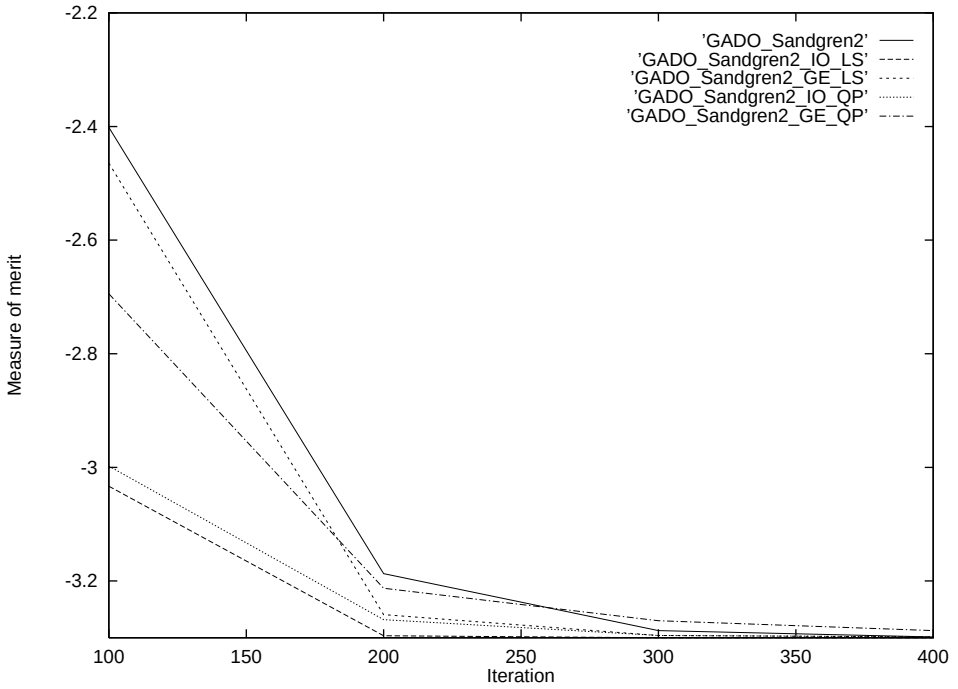

Figure 4.12: Comparison of methods for using reduced models in benchmark domain 2

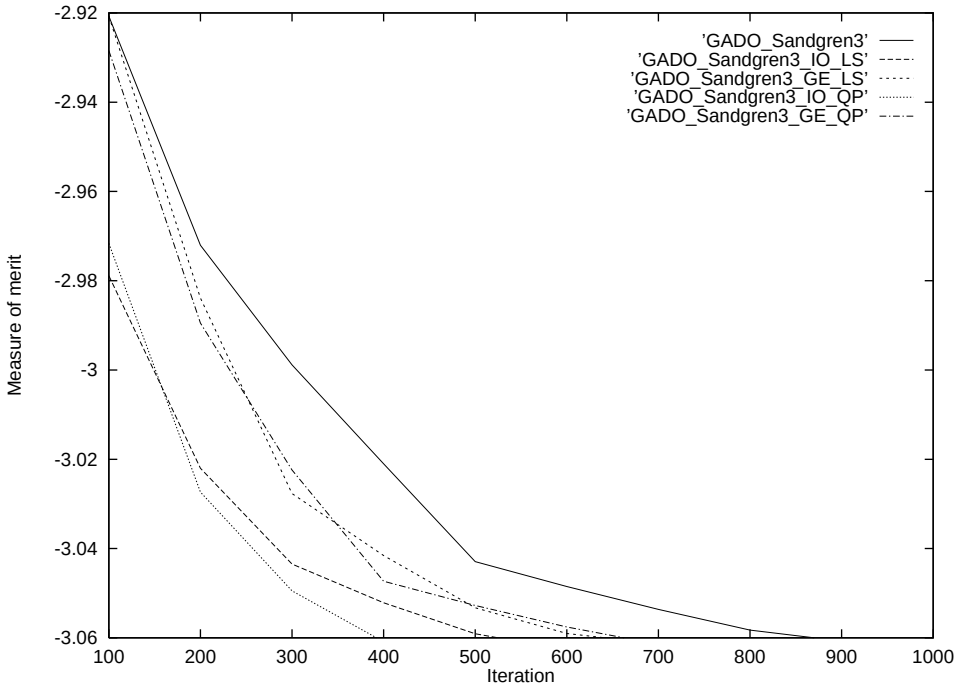

Figure 4.13: Comparison of methods for using reduced models in benchmark domain 3

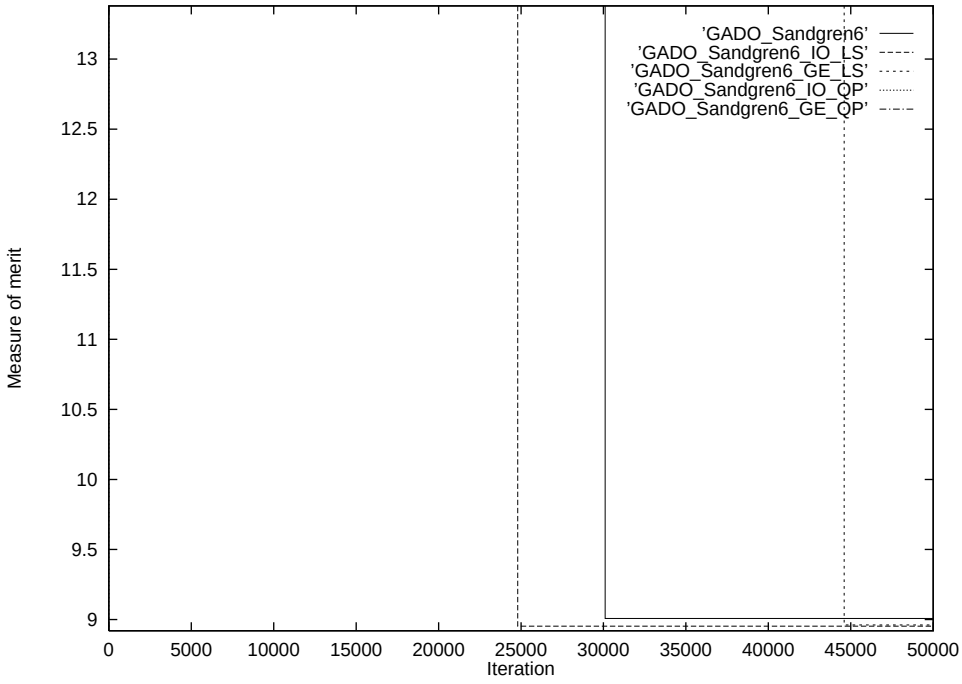

Figure 4.14: Comparison of methods for using reduced models in benchmark domain 4

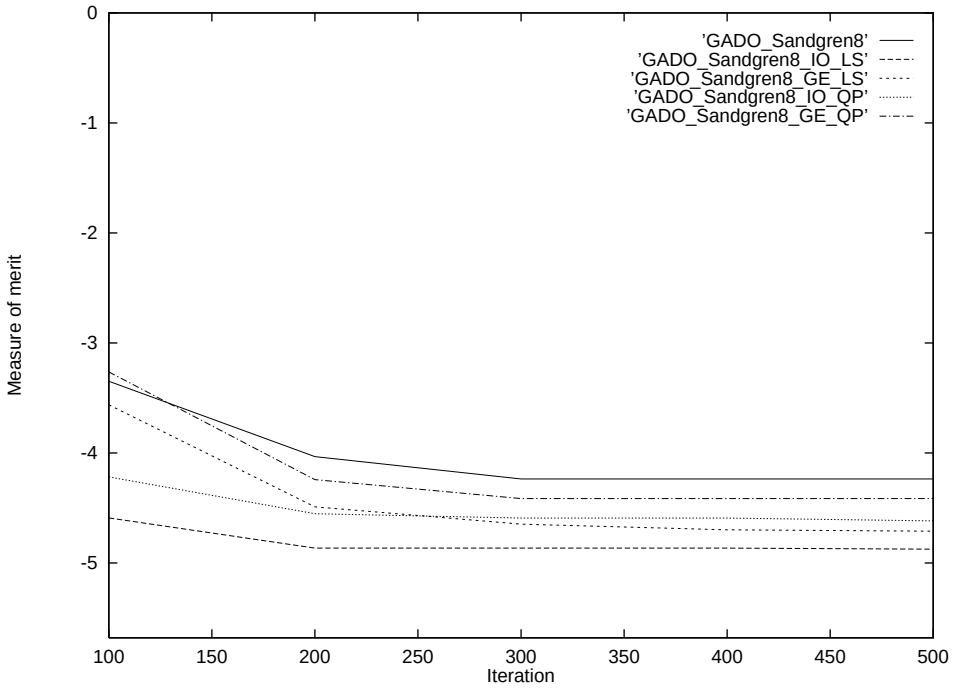

Figure 4.15: Comparison of methods for using reduced models in benchmark domain 5

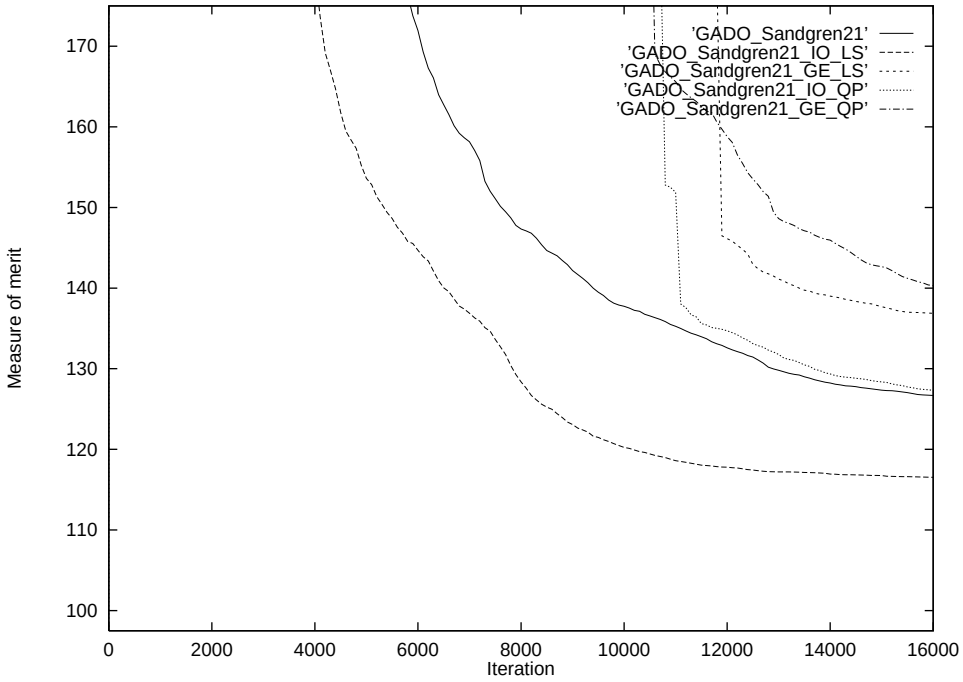

Figure 4.16: Comparison of methods for using reduced models in benchmark domain 6

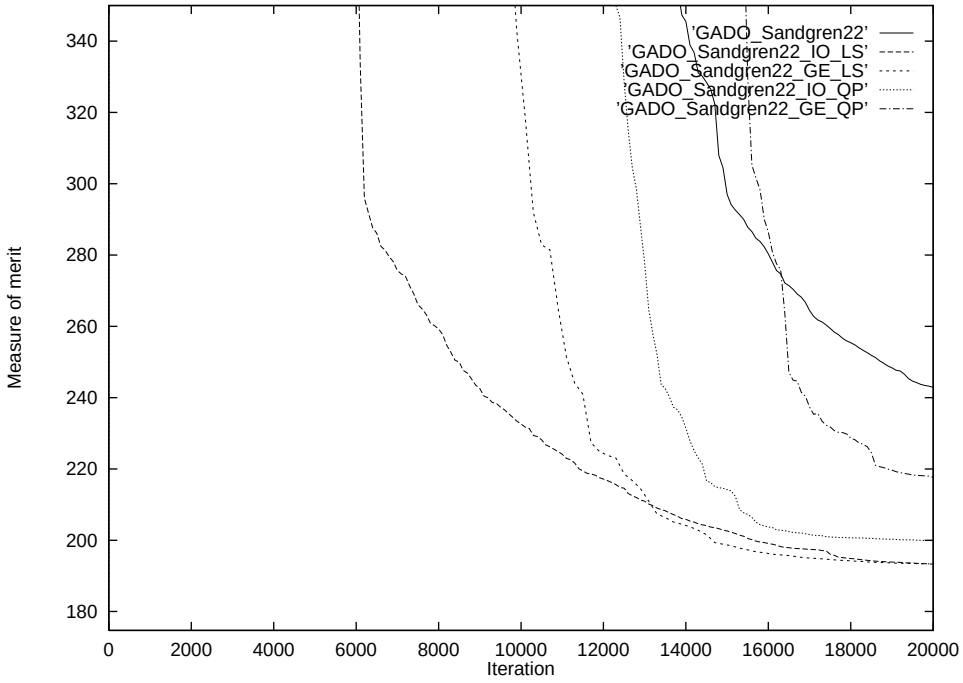

Figure 4.17: Comparison of methods for using reduced models in benchmark domain 7

show the performance with each of the four combinations as well as performance with no approximation or speedup at all (the solid lines) in the benchmark domains. Each curve in each figure shows the average of 15 runs of GADO with different random seeds. We found that in the first four benchmarks, which represent relatively easy optimization tasks, the performance differences were small. The figures show that the IO based approach did better than the GE approach using the same approximation technique (LS or QP) in most stages of most domains. The figures also show that the IO method gave the best final performance in all domains. In fact, the results with the GE approach in benchmark 6 were worse than with no speedup at all. In benchmark 3, IO with QP was the winner while in all other benchmarks the IO with LS was the winner suggesting that LS was better than QP as an approximation

method. We should also point out that in benchmark 7, in which GE appears to be doing better than IO for a segment of the optimization, we found that one of the runs did not find any feasible points but was slightly infeasible till the end. Thus, the GE performance in this domain is worse than the curve suggests.

4.4 Discussion

In this thesis, we first compare different methods for forming dynamic reduced models to speed up the search in GA-based engineering design optimization. Experiments were conducted in the domain of aircraft design optimization as well as several benchmark engineering design domains. The experiments show that the quadratic least squares approach did consistently well and was the best in the more difficult problems such as aircraft design and benchmarks 6 and 7. Another factor in favor of the least squares approach is that forming the approximations with this method is more than an order of magnitude faster than with the other methods and does not require any tuning of parameters like them. Yet another advantage of least squares methods is that the approximation is in a mathematical form (polynomial or otherwise) which could be algebraically analyzed and manipulated as opposed to the black-box results that neural networks give.

On the other hand, we did a comparison between two methods for using reduced models to speed up the search in GA-based engineering design optimization. The experiments show that the informed operators approach did consistently well and was better than the genetic engineering approach in all domains. Moreover, the genetic engineering approach called the approximate fitness function an order of magnitude more times than the informed operators approach. We believe that the reason for this result is that the reduced models used were not accurate enough for the genetic engineering approach to yield good results. The informed operators

approach on the other hand makes a much weaker assumption about the accuracy of the reduced model (all it needs to speed up the optimization is that the reduced model be a better than random predictor of the actual model). The experiments also showed that using least squares approximations with any speedup approach usually yields better results than using the neural network approximations.

Chapter 5

Summary

This thesis concerns building and using dynamic reduced models to accelerate genetic-algorithm-based design optimization. Least squares approximation, one of the traditional numerical functional approximation methods, was previously used to achieve the speed-up purpose. ANN is a powerful statistical technique to fulfill functional approximations. Theoretically, it should work as well as the least squares technique in the reduced model context. With this in mind, we integrated ANN based reduced model into GADO and tested it against the least squares based GADO. The figures in chapter four have demonstrated the comparison of three different methods. Empirical results show that the quadratic least squares method based GADO does consistently better in the aircraft design domain and most of the benchmark engineering design domains.

In essence, we make use of neural networks and quadratic least squares methods to perform function approximation. ANN can be viewed as a black box, and with the proper configuration of network structure, it can serve as a universal approximator. ANN was hoped to give more accurate predictions so that a more precise reduced model is built. However, the drawback of this technique should also be taken into consideration. As one of the statistical techniques, neural networks require a large sample set to learn the underlying relationship between inputs and outputs. The configuration of the neural network is another concern. As is known, neural networks work in a trial and error manner, which means we can only evaluate the ANN

performance by experiments and error analysis. In our work, ANN is embedded in the GADO system, which makes it more difficult to monitor the training process. It may probably be difficult for the ANN to adapt to different environments, if we keep the network structure unchanged. Our experimental results in chapter four have shown the feasibility of the ANN based reduced model. Although unable to outperform the previous least squares method based reduced model, ANN based reduce model work better than the original uninformed (without the reduced model) GADO.

Our next concern is how to use reduced models to speed up optimization. Besides informed operators, we want to try other ways to use reduced models. Genetic Engineering is another approach to take advantage of existing numerical methods, based on the knowledge provided by reduced models. Downhill-Simplex has been selected, because it is not a gradient-based technique. The results in chapter four show that the informed operator approach outperforms the Genetic Engineering approach in most test domains. The reason may be our reduced models are not accurate enough for the Genetic Engineering method to work well. The informed operators approach is based on a much weaker assumption of the accuracy of the reduced models.

The future work includes repeating the comparison of speedup approaches under different neural network models for approximation, such as radial-bases-function neural networks and multi-layer perceptrons. Another possible direction is to make the neural networks more adaptable to the environmental changes. We hope that the neural networks can be tuned in a more automatic, domain independent fashion. We may also explore ways of combining the informed-operator approach and the genetic engineering approach to achieve better performance than using any single approach. We would like to study the conditions under which each method is better. We also hope to be able to repeat the comparison in situations in which the reduced models are physical, pre-existent or somehow more accurate. Finally we may explore other

speedup approaches such as methods based on the formation and instantiation of statistical models.

Bibliography

[1] Guido Cervone, Kenneth Kaufman, and Ryszard Michalski. Experimental validations of the learnable evolution model. In *Proceedings of the 2000 Congress on Evolutionary Computation CEC00*, pages 1064–1071, 6-9 July 2000.

[2] B. Dunham, D. Fridshal, R. Fridshal, and J. North. Design by natural selection. *Synthese*, 15:254–259, 1963.

[3] D. Eby, R. Averill, W. Punch, and E. Goodman. Evaluation of injection island GA performance pn flywheel design optimization. In *Proceedings of the third Conference on adaptive computing in design and manufactoring*, 1998.

[4] Mohammed A. El-Beltagy, Prasanth B. Nair, and Andy J. Keane. Metamodeling techniques for evolutionary optimization of computationally expensive problems: Promises and limitations. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 13-17 July 1999.

[5] Scott E. Fahlmann. An empirical study of learning speed in back-propagation networks. Technical Report CMU-CS-88-162, Carnegie Mellon University, 1988.

[6] Andrew Gelsey, M. Schwabacher, and Don Smith. Using modeling knowledge to guide design space search. In *Fourth International Conference on Artificial Intelligence in Design '96*, 1996.

[7] A. J. Howell and H. Buxton. Face recognition using radial basis function neural networks. In *Proceedings of the British Machine Vision Conference*, 1996.

[8] Ian Patterson and Steve Readett. The quickprop algorithm project, 1998.

[9] D. Powell and M. Skolnick. Using genetic algorithms in engineering design optimization with non-linear constraints. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 424–431. Morgan Kaufmann, July 1993.

[10] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C : the Art of Scientific Computing*. Cambridge University Press, Cambridge [England] ; New York, 2nd edition, 1992.

[11] Khaled Rasheed. GADO: A genetic algorithm for continuous design optimization. Technical Report DCS-TR-352, Department of Computer Science, Rutgers, The State University of New Jersey, New Brunswick, NJ, January 1998. Ph.D. Thesis, http://www.cs.rutgers.edu/∼krasheed/thesis.ps.

[12] Khaled Rasheed. Guided crossover: A new operator for genetic algorithm based optimization. In *Proceedings of the Congress on Evolutionary Computation*, 1999.

[13] Khaled Rasheed. An incremental-approximate-clustering approach for developing dynamic reduced models for design optimization. In *Proceedings of the Congress on Evolutionary Computation (CEC)*, 2000.

[14] Khaled Rasheed and Haym Hirsh. Learning to be selective in genetic-algorithm-based design optimization. *Artificial Intelligence in Engineering, Design, Analysis and Manufacturing*, 13:157–169, 1999.

[15] Khaled Rasheed and Haym Hirsh. Informed operators: Speeding up genetic-algorithm-based design optimization using reduced models. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 2000.

[16] Khaled Rasheed, Xiao Ni, and Swaroop Vattam. Comparison of methods for developing dynamic reduced models for design optimization. In *Proceedings of the Congress on Evolutionary Computation (CEC'2002)*, 2002.

[17] Eric Sandgren. The utility of nonlinear programming algorithms. Technical report, Purdue University, 1977. Ph.D. Thesis.

[18] Vassili V. Toropov and Luis F. Alvarez. Application of genetic programming to the choice of a structure of global approximations. In *Genetic Programming 1998: Proceedings of the Third Annual Conference*, 1998.