Comparison of Methods for Developing Dynamic Reduced Models for Design Optimization

Khaled Rasheed and **Xiao Ni** and **Swaroop Vattam**
Department of Computer Science and Artificial Intelligence Center
The University of Georgia, Athens, GA 30605, USA
khaled@cs.uga.edu, xiaoni@ai.uga.edu, svattam@ai.uga.edu

Abstract - In this paper we compare three methods for forming reduced models to speed up genetic-algorithm-based optimization. The methods work by forming functional approximations of the fitness function which are used to speed up the GA optimization by making the genetic operators more informed. Empirical results in several engineering design domains are presented.

I. Introduction

This paper 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".

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 paper 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. [3] 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 [6]. Other approaches rely on special relations between the approximate and accurate model to develop interesting multi-level search strategies [7]. A recent approach [8] 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.

We conducted our investigations in the context of GADO [2], a GA that was designed with the goal of being suitable for use in engineering design. It uses new operators and search control strategies that target 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 [9] that saves time by avoiding the evaluation of points that are unlikely to correspond to good designs.

The remainder of this paper first presents brief descriptions of the compared methods for reduced model formation. We then present a number of experiments concerning the use of these approximation methods on one realistic engineering design task and several engineering design benchmarks. We conclude the paper with a discussion of the results and future work.

II. Reduced model formation methods

A. General framework

We conducted our investigation in the context of the framework described in detail in [10]. 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.

A.0.a 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.

A.0.b 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).

B. 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:

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

  where n is the dimension of the search space.
  We use a least square fitting routine from [11]. It works by using the normal equations method.
- **Cluster approximation functions**

[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.

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.

C. 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 [5]. 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 square error). 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 [5].

D. 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 [4]. Quickprop learns much faster than the standard back-propagation but also has more parameters that have to be fine-tuned. 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. After examining the two approaches, we have found that the two-network-one-output approach is better. The default number of hidden nodes was set to 26, and the maximum training epochs was 800. We have used the maximum growth factor of 1.75, as was recommended in Fahlman's empirical report [4]. As with the RBF network, we introduced a mechanism for avoiding over-training in this network.

The Quickprop algorithm was implemented and integrated into GADO so as to use both the global and the cluster approximation models. We form two types of 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.

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 [12] where we demonstrate their use 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.

III. Experimental results

To compare the performance of the different approximation methods we compare GADO with reduced-model-based informed operators based on each approximation 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: one domain from real tasks in aerodynamic design, plus seven others from an existing set of engineering design benchmarks [1], [13].

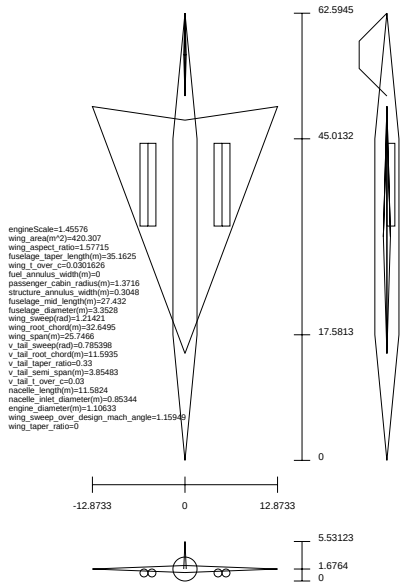

Fig. 1. Supersonic transport aircraft designed by our system

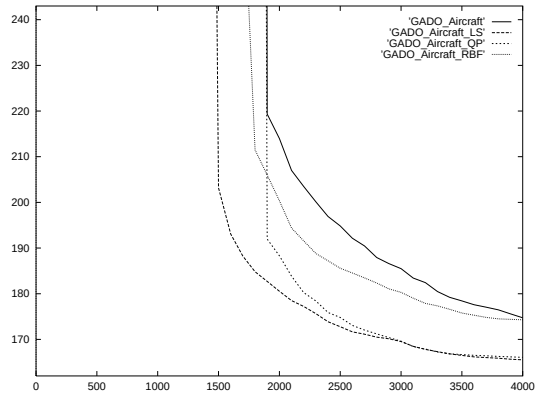

Fig. 2. Comparison of average performance in application domain 1 (aircraft design)

A. Application domain 1: Supersonic transport aircraft design domain

A.1 Domain description

Our first domain concerns the conceptual design of supersonic transport aircraft. We summarize it briefly here; it is described in more detail in [14]. Figure 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. In summary, the problem has 12 parameters and 37 inequality constraints and 0.6% of the search space is evaluable.

A.2 Experiments and results

Figure 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 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 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.

B. Benchmark engineering design domains

In order to further compare the three approximation methods, we compared their performance in several benchmark engineering design domains. These domains were introduced by Eric Sandgren in his Ph.D. thesis [1] in which he 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 [13]. A detailed description of these domains is given in [1].

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 3 through Figure 9 show the performance with each of the three approximation

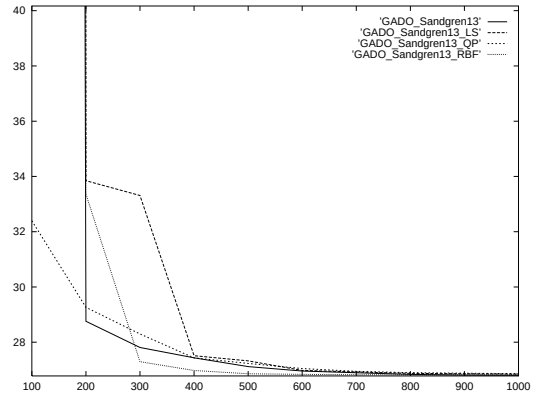

Fig. 3. Comparison of average performance in benchmark domain 1

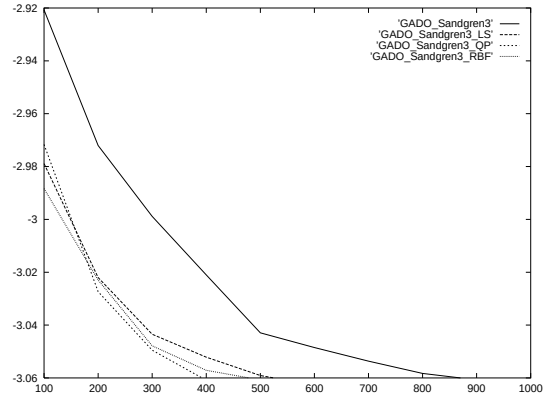

Fig. 5. Comparison of average performance in benchmark domain 3

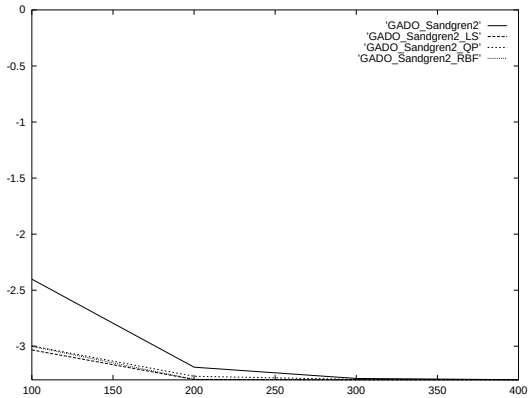

Fig. 4. Comparison of average performance in benchmark domain 2

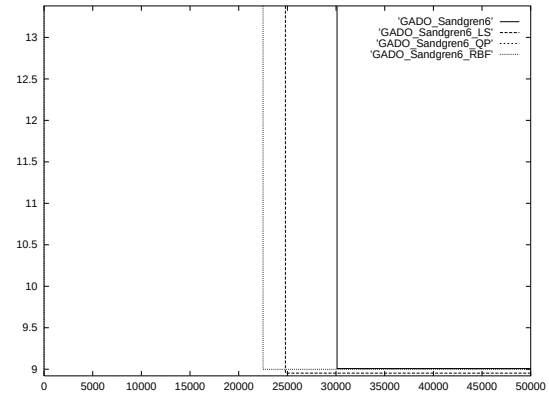

Fig. 6. Comparison of average performance in benchmark domain 4

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.

IV. Final Remarks

This paper has presented a comparison between 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.

In the future, we intend to explore different neural network models for approximation, such as multi-layer perceptrons. Finally, we want to investigate the least squares approach more carefully as it proved to be the best so far. We would like to explore the use of non-

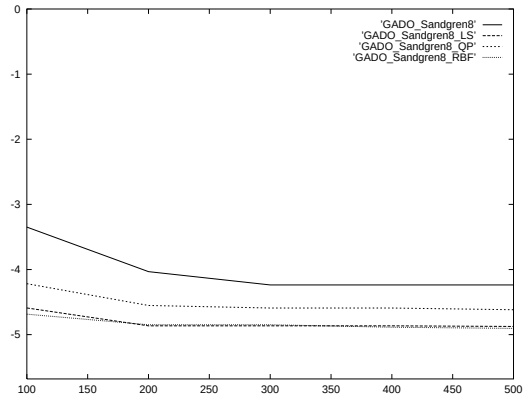

Fig. 7. Comparison of average performance in benchmark domain 5

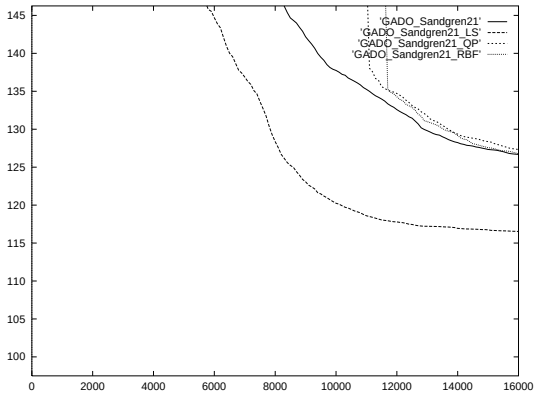

Fig. 8. Comparison of average performance in benchmark domain 6

polynomial basis functions and spline functions for approximation.

Acknowledgments

This research was funded in part 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.

References

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

[2] 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.

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

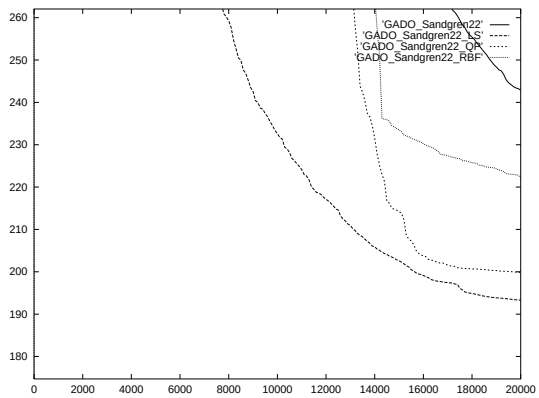

Fig. 9. Comparison of average performance in benchmark domain 7

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

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.