Comparison of Methods for Using Reduced Models to Speed Up Design Optimization

Khaled Rasheed
Computer Science Department
The University of Georgia
Athens, GA 30605, USA
khaled@cs.uga.edu
Phone:(706)542-3444

Swaroop Vattam
Artificial Intelligence Center
The University of Georgia
Athens, GA 30605, USA
svattam@ai.uga.edu
Phone:(706)542-0358

xiao Ni
Artificial Intelligence Center
The University of Georgia
Athens, GA 30605, USA
xiaoni@ai.uga.edu
Phone:(706)542-0358

Abstract

In this paper we compare two 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. One method speeds up the optimization by making the genetic operators more informed. The other method speeds up the optimization by genetically engineering some individuals instead of using the regular Darwinian evolution approach. Empirical results in several engineering design domains are presented.

1 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. We compare two methods for improving the GA's performance. One is the idea of informed operators (IO) presented in [7] which uses the approximations to make the GA operators such as crossover and mutation more effective. The other is a variation of the genetic engineering (GE) idea presented in [8] which improves 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.

The use of reduced models to save time in evolutionary optimization dates all the way back to the sixties. Dunham et al. [4] 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 [9]. Other approaches rely on special relations between the approximate and accurate model to develop interesting multilevel search strategies. A notable class of such methods [10] 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 [11] 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 [3, 12], 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 [12] 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 use for speedup. The paper then presents brief descriptions of the approximation methods used to form the reduced models. 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.

2 Reduced-model-based speedup methods

We compare two methods for improving the GA's performance. One is the idea of informed operators presented in [7] and the other is a variation of the genetic engineering idea presented in [8]. The remainder of this section describes these two approaches in more detail.

2.1 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 [3] 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 [6] 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 [6]. 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.

2.2 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 [13], is used because of its lesser commitment to 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 favour the IO approach anyway.

3 Reduced model formation methods

In this section we briefly describe the methods used to create the approximate models which the speedup methods use. More detailed descriptions of these methods can be found in [14, 15].

3.1 General framework

We conducted our investigation in the context of the framework described in detail in [14]. We provide only a brief description here. The 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 a smaller sample may be used to keep the overhead 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. We use clustering because it makes it possible to fit discontinuous and complicated surfaces with simpler surfaces such as quadratic approximations.

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).

3.2 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:

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

  where n is the dimension of the search space and x_i is design variable number i. We use a least square fitting routine from [13] which works by using the normal equations method to fit the a_i values.

- **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.

3.3 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]. 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 [2] of Quickprop, translated by Terry Regier from Fahlman's original Lisp version.

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. We also introduced a mechanism for avoiding over-training in this network.

The Quickprop algorithm was implemented and integrated into GADO so as to generate 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 approximation.

4 Experimental results

To compare the performance of the different speedup methods we compared GADO with reduced-model-

[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.

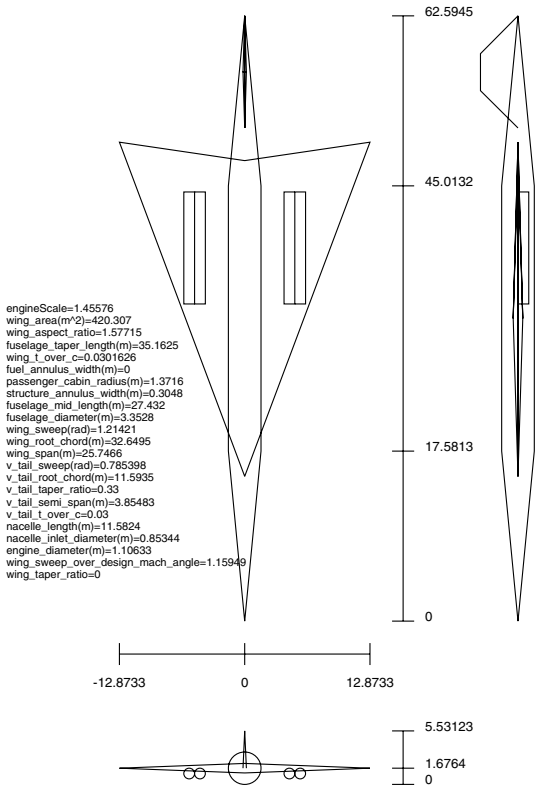

Figure 1: Supersonic transport aircraft designed by our system (dimensions in feet)

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: one domain from real tasks in aerodynamic design, plus seven others from an existing set of engineering design benchmarks [1].

4.1 Application domain 1: Supersonic transport aircraft design domain

4.1.1 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 [16]. 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

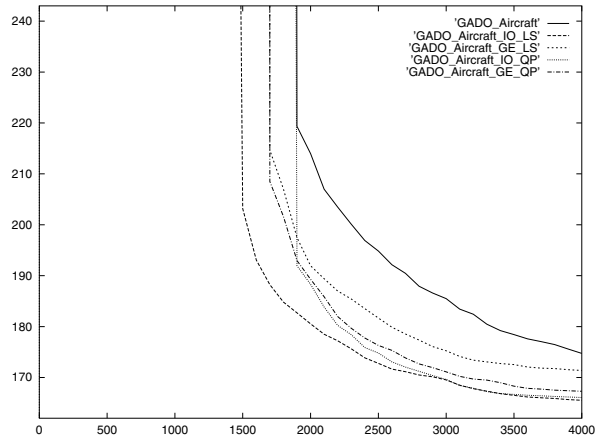

Figure 2: Comparison of average performance in application domain 1 (aircraft design)

measure of merit commonly used in the aircraft industry at the conceptual design stage. Takeoff mass is the sum of fuel mass, which provides a rough approximation of the operating cost of the aircraft, and "dry" mass, which provides a rough approximation of the cost of building the aircraft. In summary, the problem has 12 parameters and 37 inequality constraints. 0.6% of the search space is evaluable. No statistics exist regarding the fraction of the search space that is feasible because it is extremely small.

4.1.2 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 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 2 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 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 speedup methods affect the number of calls to the objective function in problems where the informed operators or genetic engineering overhead is minuscule compared to the runtime of each objective function evaluation, as was the case here. This

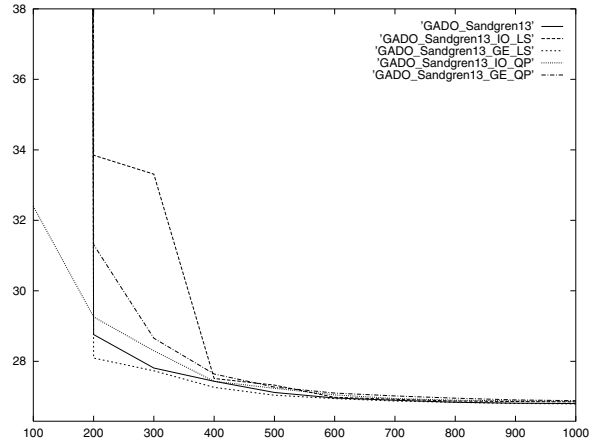

Figure 3: Comparison of average performance in benchmark domain 1

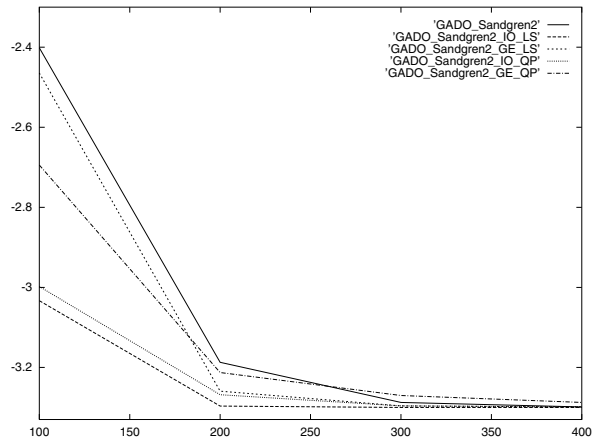

Figure 4: Comparison of average performance in benchmark domain 2

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 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.

4.2 Benchmark engineering design domains

In order to further compare the two speedup 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 [17]. A detailed description of these domains is given in [1].

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 3 through Figure 9 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

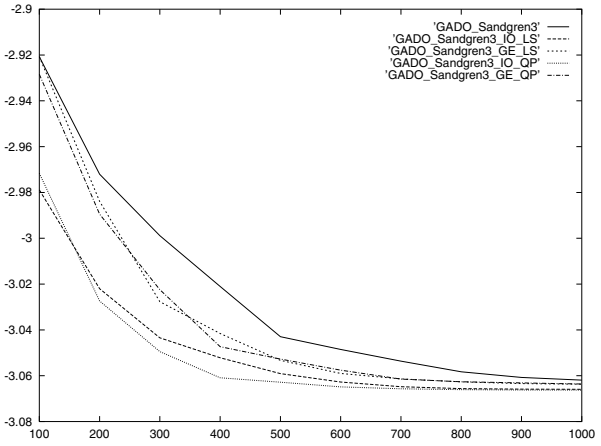

Figure 5: Comparison of average performance in benchmark domain 3

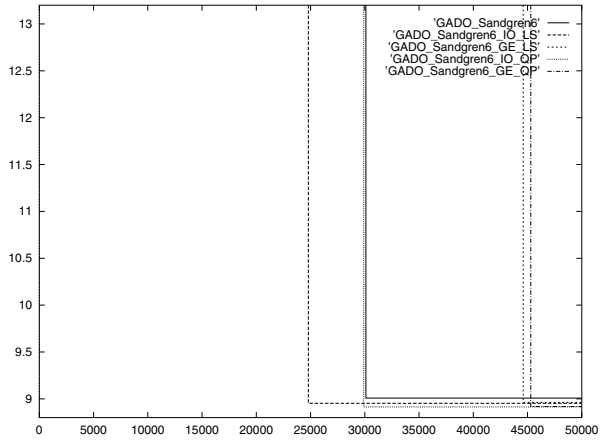

Figure 6: Comparison of average performance in benchmark domain 4

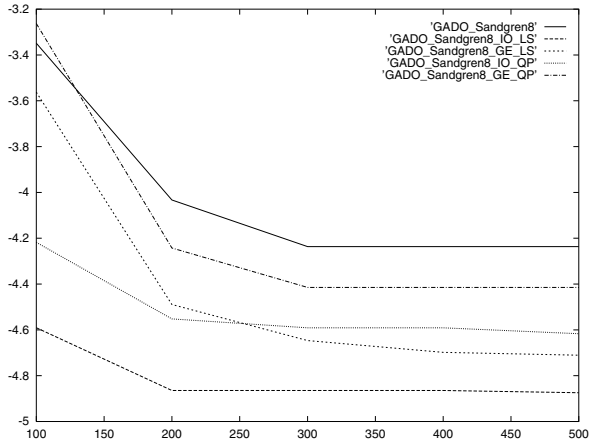

Figure 7: Comparison of average performance in benchmark domain 5

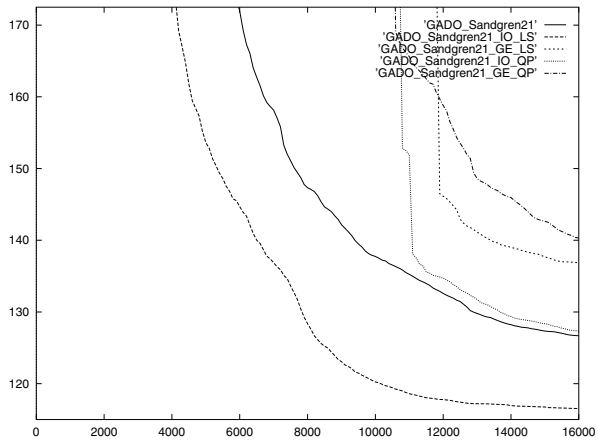

Figure 8: Comparison of average performance in benchmark domain 6

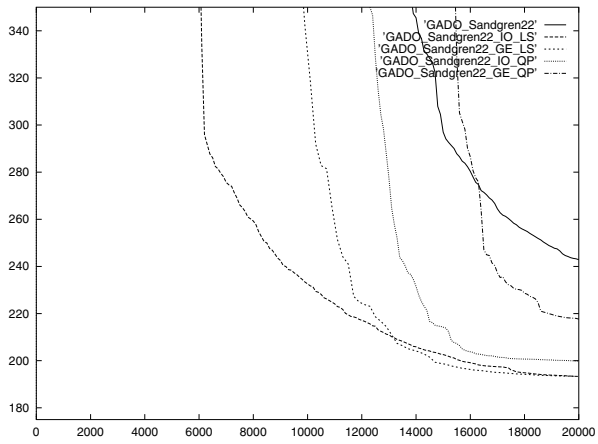

Figure 9: Comparison of average performance in benchmark domain 7

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.

5 Final Remarks

This paper has presented a comparison between two methods for using 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 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.

In the future, we intend to repeat the comparison of speedup approaches under different neural network models for approximation, such as radial-bases-function neural networks and multi-layer perceptrons. We also intend to explore ways of combining the informed-operator approach and the genetic engineering approach to achieve better performance than using any single approach. 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 but unfortunately we do not have access to such domains at this time. Finally we intend to explore other speedup approaches such as methods based on the formation and instantiation of statistical models.

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

[3] 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.

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

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.