Informed operators: Speeding up genetic-algorithm-based design optimization using reduced models

Khaled Rasheed
Computer Science Department
Rutgers University
New Brunswick, NJ 08903
krasheed@cs.rutgers.edu
Phone:(732)445-4379

Haym Hirsh
Computer Science Department
Rutgers University
New Brunswick, NJ 08903
hirsh@cs.rutgers.edu
Phone:(732)445-4176

Abstract

In this paper we describe a method for improving genetic-algorithm-based optimization using informed genetic operators. The 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 a reduced model, then take the best to be the result of the mutation. The proposed method is particularly suitable for search spaces with expensive evaluation functions, such as arise in engineering design. Empirical results in several engineering design domains demonstrate that the proposed method can significantly speed up the GA optimizer.

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".

One of the major problems faced in the application of GAs (or any optimization technique for that matter) to such problems is that the simulator will often take a non-negligible amount of time to evaluate a point. Engineering simulation run time can range often from a fraction of a second to, in some cases, many days. 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 presents a modification of GAs specifically intended to improve performance in realistic engineering design domains of this sort. The idea is to replace any point where a purely random decision is made into one that is influenced by the assessments of the reduced model. In typical GAs, the initial population is formed by generating a number of individuals equal to the population size where the design variables of each individual are selected uniformly randomly over their respective ranges. Similarly, crossover entails a random decision such as picking the crossover point(s). Mutation is usually done by selecting a random point in the vicinity of the point to be mutated. We still generate a random initial population and do crossover and mutation stochastically. The modification we propose is to generate several purely random candidates at every step of initial population generation or crossover or mutation, then rank these random candidates using some reduced model. The best of the random candidates according to the reduced model is taken to be the final outcome for initialization, crossover or mutation. If the time spent in ranking the random alternatives using the reduced model is negligible compared to the time it takes to run a simulator, this policy could result in excellent speed-up.

The use of reduced models to save time in evolutionary optimization dates all the way back to the sixties. Dunham et al. [1] 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 [12]. 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 [13, 2] 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. Some approaches [3] use both an accurate/expensive and a cheaper/approximate model interchangeably during the search by relying on intelligent techniques to decide when it is safe to rely on the cheaper model.

We observed that most of the above efforts make a strong assumption regarding how accurately the approximate model actually approximates the more accurate model. Substituting cheap evaluations for more accurate ones is very risky and could mislead the search in ways that cannot be recovered from in later stages. The assumption that finite element computations with fewer elements can accurately predict computations with more elements does not hold in many aerodynamic and especially inlet design domains that we have seen. In contrast, this paper presents an approach in which all that is required to save time is for the cheap model to be a better-than-random approximation of the accurate one. Our approach is also simple and easy to add to an existing genetic algorithm.

We conducted our investigations in the context of GADO [7, 10], 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). A steady state GA model is used, in which operators are applied to two parents selected from the elements of the population via some selection scheme, one offspring point is produced, then an existing point in the population is replaced by the newly generated point via some replacement strategy. Here selection was performed by rank because of the wide range of fitness values caused by the use of a penalty function. The replacement strategy used here is a crowding technique, which takes into consideration both the fitness and the proximity of the points in the GA population. The GA stops when either the maximum number of evaluations has been exhausted or the population loses diversity and practically converges to a single point in the search space. 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 [10] that saves time by avoiding the evaluation of points that are unlikely to correspond to good designs.

The remainder of this paper first presents a detailed description of the informed operators. We then present a number of experiments concerning the use of the informed operators on two realistic engineering design tasks and two engineering design benchmarks. We conclude the paper with a discussion of future work.

2 Informed operators

In many optimization tasks to date, the functions to optimize were analytical expressions that take a negligible amount of time to compute. In the case of realistic engineering design, however, the optimization function is more often than not a computationally expensive piece of code such as a simulator. We encountered cases in which one evaluation of the optimization function takes CPU hours on a powerful workstation. Consequently, it is very desirable to use reduced models, which are usually orders of magnitude faster than the actual simulator, to cut down the number of calls to the actual simulator.

Reduced models can be physical, such as models relying on simpler physical equations like one-dimensional gas dynamics instead of two or three-dimensional, or numerical approximations such as response surfaces induced using some input output pairs evaluated by the original expensive model.

Regardless of how the reduced models come to existence, it is important to develop an efficient method to use them to speedup the search. The ultimate goal is to find solutions with the same quality as the case in which the actual model can be used very liberally, with significantly decreased reliance on the model.

We use 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 [7] 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 [8] 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 [8]. 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.

In most of the experiments described below, we used reduced models that are acquired on-line as the optimization progresses. The technique for computing these dynamic reduced models is described in further detail elsewhere [9]. The idea is to maintain a large sample of the points encountered in the course of the optimization divided into dynamic clusters and periodically compute least squares approximations for the measure of merit and the constraint violations for each large cluster as well as the whole sample. To compute the approximate fitness of a point, the point is first classified using a distance weighted k-nearest-neighbor vote to either feasible, infeasible-evaluable or unevaluable (to be defined shortly). Then, depending on the classification and cluster of the point, the appropriate approximation functions are used to approximate the fitness.

3 Experimental results

To demonstrate the utility of the informed operators we compared GADO with its default setup to a modified version where the informed operators replaced the pure random operators but with all other parameters intact. We then compared the two systems in several domains: two domains from real tasks in aerodynamic design, plus two others from an existing set of engineering design benchmarks [11].

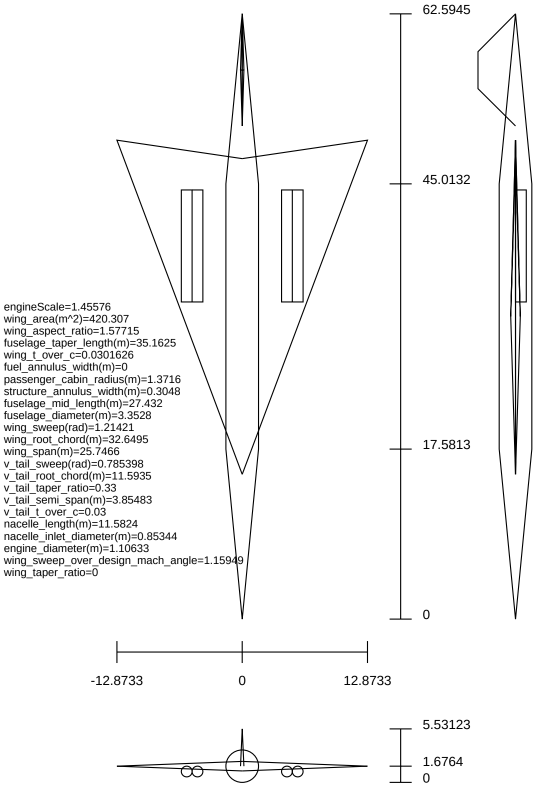
62.5945
45.0132
17.5813
0
-12.8733
0
12.8733
5.53123
1.6764
0
engineScale=1.45576
wing_area(m^2)=420.307
wing_aspect_ratio=1.57715
fuselage_taper_length(m)=35.1625
wing_t_over_c=0.0301626
fuel_annulus_width(m)=0
passenger_cabin_radius(m)=1.3716
structure_annulus_width(m)=0.3048
fuselage_mid_length(m)=27.432
fuselage_diameter(m)=3.3528
wing_sweep(rad)=1.21421
wing_root_chord(m)=32.6495
wing_span(m)=25.7466
v_tail_sweep(rad)=0.785398
v_tail_root_chord(m)=11.5935
v_tail_taper_ratio=0.33
v_tail_semi_span(m)=3.85483
v_tail_t_over_c=0.03
nacelle_length(m)=11.5824
nacelle_inlet_diameter(m)=0.85344
engine_diameter(m)=1.10633
wing_sweep_over_design_mach_angle=1.15949
wing_taper_ratio=0

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

3.1 Application domain 1: Supersonic transport aircraft design domain

3.1.1 Domain description

Our first domain concerns the conceptual design of supersonic transport aircraft. We summarize it briefly here; it is described in more detail elsewhere [4]. 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. 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. A complete mission simulation requires about 0.2 CPU seconds on a DEC Alpha 250 4/266 desktop workstation.

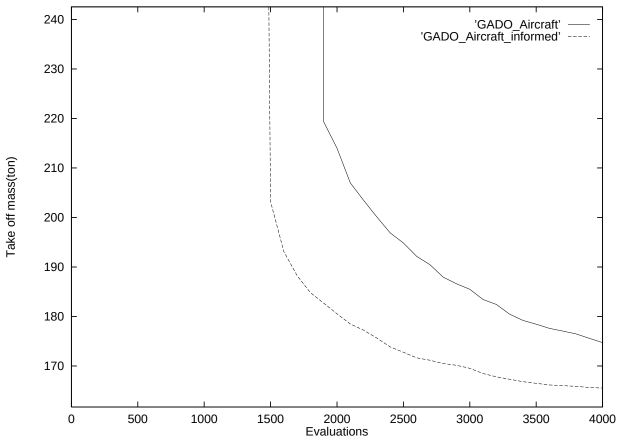

Figure 2: Effect of informed operators on average performance in application domain 1 (aircraft design)

The aircraft simulation model used is based on both implicit and explicit assumptions and engineering approximations and since it is being used by a numerical optimizer rather than a human domain expert, some design parameter sets may correspond to aircraft that violate these assumptions and therefore may not be physically realizable even though the simulator does not detect this fact. We refer to these designs as *infeasible points*. For this reason a set of constraints has been introduced to safeguard the optimization process against such violations. We also have the notion of *unevaluable points*. These are points that represent designs that violate the model assumptions so much that the simulator cannot complete the simulation process to produce any significant information. For such points a very large fictitious takeoff mass is generated as the value of the objective function.

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. A detailed description of the experiments comparing GADO to other optimizers in this domain can be found elsewhere [7]. Those experiments strongly demonstrated the merit and competitiveness of GADO compared to both stochastic and non-stochastic search methods. Here our focus is on studying the effect of the informed operators on performance.

3.1.2 Experiments and results

Figure 2 demonstrates the utility of the informed operators in domain 1 (aircraft design). The figure shows the average of 15 runs of GADO both with and without the informed operators. A few parameters were set to

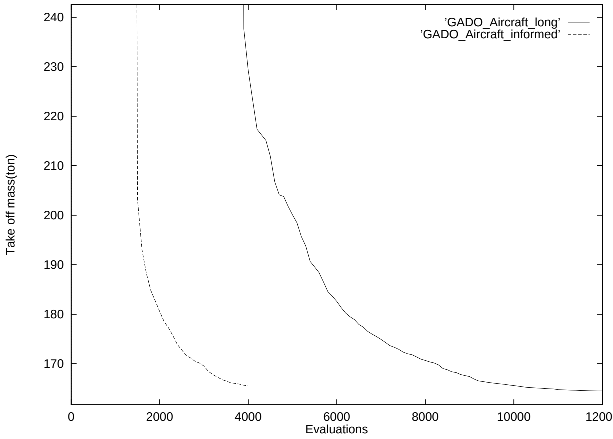

Figure 3: Speed-up comparison in application domain 1 (aircraft design). Note the horizontal scale difference from the previous figure.

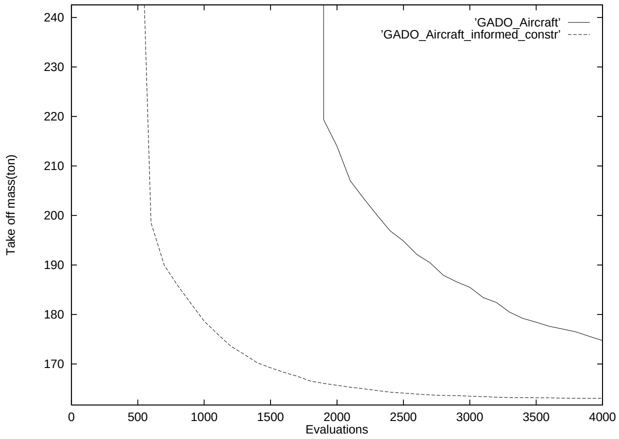

Figure 4: Effect of constraint based informed operators on average performance in application domain 1 (aircraft design)

more aggressive values relative to previously published defaults [7] to speedup the experiments and show how the informed operators can permit fast convergence with little or no loss in quality. For example, the population size was set to 40 rather than its default value of 120 (10 times the dimension). However, all parameters were kept at the same values in all experiments with or without informed operators.

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 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 informed operators improved the performance in all stages of the search. The feasible region was reached faster with the informed operators.[1]

Figure 2 demonstrates how GADO with the informed operators found much better designs than GADO without the informed operators. However, we know from past experience that GADO was capable of finding excellent designs even without the informed operators, provided enough time was given and more conservative settings were used (larger population for example). Figure 3 compares GADO with informed operators given 4000 iterations to GADO with its original configuration [7, 10] and given 12000 iterations. We find that GADO with the informed operators after 4000 iterations dominates GADO without informed operators and with the original GADO configuration until iteration 10000, where by dominance we mean that the 15 informed GADO runs at iteration 4000 had a better average takeoff mass and lower standard deviation than the corresponding original GADO runs at 10000 iterations.

The informed operators in these results are based on reduced models formed online, during the course of an optimization. We also performed some additional experiments in this domain to understand the case where more accurate pre-existing reduced models can be used. In these experiments we generated a "virtual" reduced model that ran the model, but took only its results concerning constraint violations (penalties that appear in the fitness function), pretending that the run-time was non-existent. Even though the implementation we have of the aircraft simulator returns both the measure of merit and the constraints at the end, we believe that with some programming effort it can be made to compute most constraints up front. The results using this virtual reduced model are shown in Figure 4. The figure again demonstrates how a reduced model, even if only approximate, can be used with informed operators to great advantage.

[1] The almost vertical leading parts of the curves are where the feasible region was reached. These parts are so steep because of the large penalty term that goes to zero for feasible points.

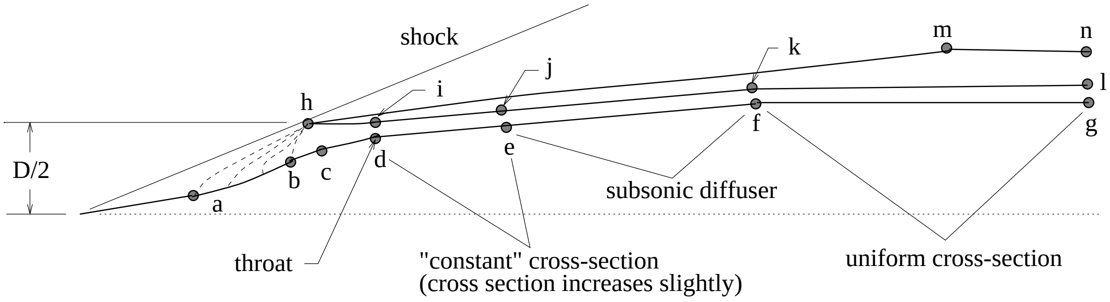

Figure 5: Supersonic missile inlet geometry model

3.2 Application domain 2: Supersonic cruise missile inlet domain

3.2.1 Domain description

Our second domain concerns the design of inlets for supersonic and hypersonic missiles. We summarize it briefly here; it is described in more detail elsewhere [14].

The missile inlet designed is an axisymmetric mixed compression inlet that cruises at Mach 4 at 60,000 feet altitude. Minimum manufacture cost for this inlet is critical, and therefore, techniques such as boundary layer bleed and variable geometry are not used — the performance of the inlet thus relies solely on the aerodynamic design of the rigid geometry, such as the extent of external and internal compression, contraction ratio, inlet start throat area, throat location, shock train length, and divergence of subsonic diffuser.

Figure 5 shows the model of the missile geometry which is composed of six fixed parameters and eight design parameters that the optimizer varies over continuous ranges. The missile inlet is axisymmetric. The goal of the optimization is to maximize the total pressure recovery, a quantity that is commonly used to measure the performance of inlets.

The simulator used in this domain is a program called “NIDA” which was developed at United Technology Research Center (UTRC) as an inlet analysis/design tool [5]. It uses a 1D aerodynamic model with the method of characteristics for the supersonic part upstream of the throat, and empirical correlations based on experimental data downstream of the throat for the region of the terminal shock wave/turbulent boundary layer interaction and sub-sonic diffuser. A complete NIDA run requires about 6 CPU seconds on a DEC Alpha 250 4/266 desktop workstation. Unfortunately, NIDA suffers from a number of serious shortcomings. There are numerous small discontinuities in the function it computes and in its first derivative, and there are numerous unevaluable points that cause NIDA to crash or print an error message. These discontinuities are sometimes in the middle of regions of good designs.

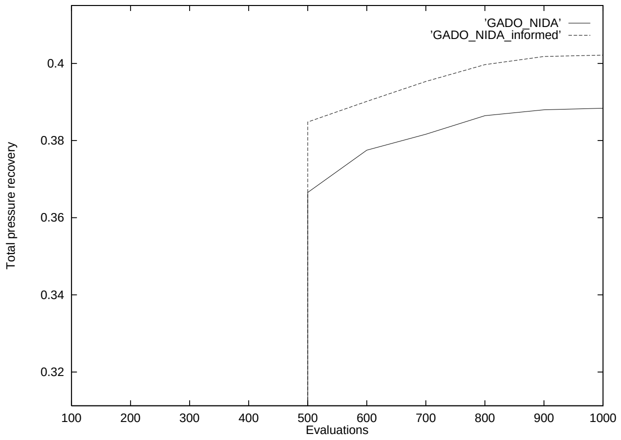

Figure 6: Effect of informed operators on average performance in application domain 2 (missile design)

In summary, the problem has eight parameters and 20 inequality constraints. 3% of the search space is evaluable and 0.147% is feasible.

3.2.2 Experiments and results

Figure 6 demonstrates the utility of the informed operators in application domain 2 (missile design). The figure shows the average of 10 runs of GADO both with and without the informed operators.[2] This time it is a maximization problem so higher points on the curve represent better designs. The informed operators had a very significant positive effect on performance in this domain.

[2] In fact one of the 10 runs without informed operators did not find any feasible points. We substituted the run, among the other nine, which ended in the worst feasible final result for that run in order to plot the curve. Thus the true performance without the informed operators is worse than the figure suggests.

Table 1: Description of benchmark domains

Domain	Sandgren	*Dim.*	**Constraints**		best
No.	*No.*		*inequ.*	*equ.*	f
1	21	13	13	0	97.5
2	22	16	19	0	174.7

As with the aircraft design domain above, we know from past experience that GADO with its original configuration that does not include the informed operators was capable of reaching the optimal region given more time and more conservative settings. A similar qualitative improvement was observed here as in the aircraft domain.

3.3 Benchmark engineering design domains

In order to further define the scope of applicability of the proposed approach, we examined its performance in two benchmark engineering design domains. These domains were introduced by Eric Sandgren in his Ph.D. thesis [11] 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. One of the recent experiments involving these domains was reported by Powell [6], in which a GA package called OOGA and a numerical optimization package called NumOpt were compared to each other in 10 of Sandgren's domains. The 10 domains were a representative sample of the original 30. We ran experiments in the most challenging two of these 10 domains. Both were minimization problems.

Some properties of the benchmarks used are summarized in Table 1. The second column of the table shows the problem numbers as they appeared in Sandgren's thesis. The third column shows the problem dimensions (i.e., the number of design variables in each problem). The fourth and fifth columns show the number of inequality and equality constraints respectively. The sixth column shows the best known optima of the problems. A detailed description of these domains is given by Sandgren [11].

For each problem GADO was run 15 times using different random starting populations. The experiments were with and without the informed operators, with all other parameters kept the same. Figure 7 and Figure 8 demonstrate the utility of the informed operators in benchmark domain 1 (Sandgren's problem 21) and benchmark domain 2 (Sandgren's problem 22) respectively. Each figure shows the average of 15 runs of GADO both with and without the informed operators. All other parameters were kept the same. The figures show that the informed operators improved the performance in all stages of the search in both domains. The feasible regions were reached faster with the informed operators. The final performance at the end of the search was significantly better with the informed operators in both cases.

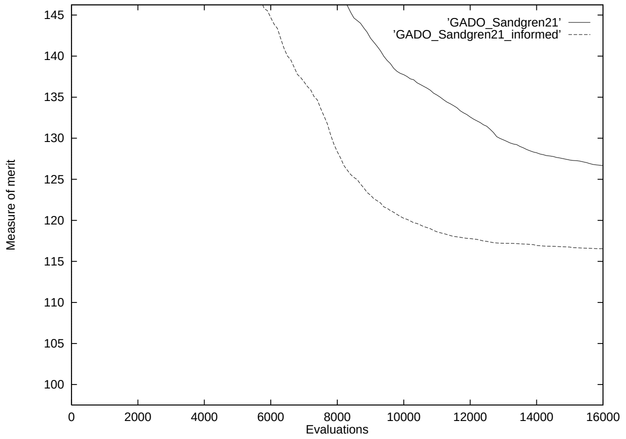

Figure 7: Effect of informed operators on average performance in benchmark domain 1 (Sandgren's problem 21)

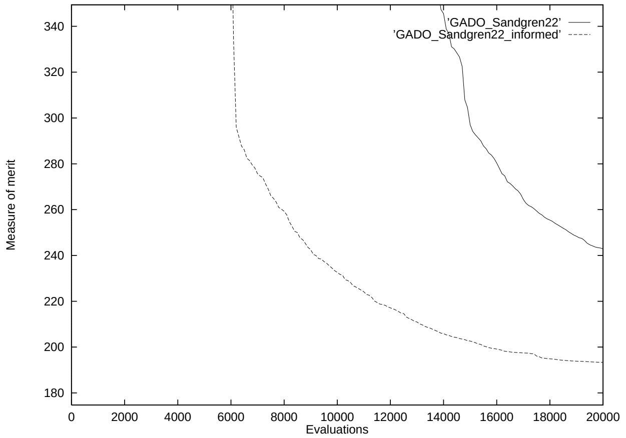

Figure 8: Effect of informed operators on average performance in benchmark domain 2 (Sandgren's problem 22)

4 Final Remarks

This paper has presented a modification of the GA for realistic engineering design search spaces that is based on using reduced models through informed genetic operators to speed up the search. Experimental results demonstrated the merit of the proposed modification in the domains of aircraft design optimization and missile inlet design as well as two benchmark engineering design domains.

We intend to further investigate ways to replace randomness with reduced model information. Genetic operators are only one way of doing this. Other ways include, for example, intelligent search control methods replacing GADO's existing search control methods [10]. We would also like to explore intelligent methods for deciding how much time to spend on the use of the reduced model. We would like to explore approaches that dynamically adjust the number of informed mutations and crossovers per iteration based on the recent accuracy of the cheap model at every stage of the search. Finally, in the cases where the reduced models are sufficiently fast, we can investigate the use of systematic optimization with the reduced model rather than just generating a number of random candidates and ranking them using the reduced models.

Acknowledgments

This research is part of 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, and we thank our colleagues on this project for their helpful suggestions during the course of this work.

References

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

[2] 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.

[3] Mohammed A. El-Beltagy and Andy J. Keane. Topographical mapping assisted evolutionary search for multilevel optimization. In *Proceedings of the Congress on Evolutionary Computation*, 1999.

[4] 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.

[5] M. Haas, R. Elmquist, and D. Sobel. NAWC Inlet Design and Analysis (NIDA) Code, Final Report. UTRC Report R92-970037-1, 1992.

[6] 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.

[7] 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.

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

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

[10] 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.

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

[12] 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.

[13] Harish D. Vekeria and Ian C. Parmee. The use of a co-operative multi-level CHC GA for structural shape optimization. In *Fourth European Congress on Intelligent Techniques and Soft Computing - EUFIT'96*, 1996.

[14] G.-C. Zha, Don Smith, Mark Schwabacher, Khaled Rasheed, Andrew Gelsey, and Doyle Knight. High performance supersonic missile inlet design using automated optimization. In *AIAA Symposium on Multidisciplinary Analysis and Optimization '96*, 1996.