GENERATING AND USING REDUCED MODELS TO IMPROVE GENETIC ALGORITHM-BASED DESIGN OPTIMIZATION

PI: Khaled Rasheed
Co-PI: Haym Hirsh
Participating Academic Institutions: Rutgers University
Topic: Advanced Computational Science

Project Summary

The process of designing complex artifacts has traditionally been done manually, exploiting the creativity and skills of human experts. The area of simulation-based design places computer systematicity where human expertise is typically employed, appealing to the same complex and costly computer codes that humans use to evaluate possible artifacts, only driven instead by suitably designed computer-based optimization methods. Our past work, and the work of many researchers have repeatedly demonstrated the promise and strength of systematic simulation-based design optimization. Simulation-based design provides the opportunity for remarkable improvements of design quality at a small fraction of the time it used to take for manual design.

One of the key distinctions between simulation-based design optimization and other forms of numerical optimization is the large time complexity of the simulations. In domains where a single evaluation can take hours, designing an artifact can require hundreds if not thousands of these expensive simulations to arrive at an attractive (and hopefully optimal) design. Our work thus far has been on the design of optimization methods – most notably, the GADO genetic-algorithm (GA) system ¡www.cs.rutgers.edu/ krasheed/gado.html¿ – that focus on decreasing the number of calls to the simulation codes and obtaining as much leverage as possible from each call.

This project pushes in a different direction, to generate and use reduced models that can more efficiently, albeit less accurately, evaluate the merit of an artifact. The use of response-surface and function-approximation methods is an obvious solution to the problem of costly evaluation codes. Indeed, a considerable literature exists on the use of response surface methods for engineering design. However, there are considerable deficiencies with the previous attempts. For example, real simulation codes often force us to confront "unevaluable" points – points in the design space that push the simulator beyond its implicit assumptions and models, often causing the simulator to crash and thereby providing no information back to the optimizer. One of the key problems intended to investigate in this project is the construction of effective approximations in the presence of unevaluable points and similar pathologies found in realistic simulation codes. Further, most research in this area has concentrated on building response surfaces "off line", using a small number of true function evaluations and then searching the resulting approximate space with little or no subsequent interaction with the true simulator or analysis code. It is also intended to explore more dynamic approaches that couple response-surface generation with the optimization method itself.

The second part of this project is to develop optimization methods that can efficiently and effectively use reduced models in their search for better designs. The reduced models can be learned as discussed above, or may already exist in some domains. One promising direction that is already being explored in initial work is to use the reduced model any place where a stochastic decision is typically made – such as mutation in a GA, replacing randomness with the hopefully more intelligent decision based on the reduced model by randomly generating several mutations and then selecting the best of them according to the reduced model.

1 Introduction

This proposal 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. We have seen domains in which more than 99% of the space is like this.
- 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 proposal presents potential modifications of GAs specifically intended to improve performance in realistic engineering design domains of this sort.

The use of reduced models to save time in evolutionary optimization dates all the way back to the sixties. [Dunham *et al.* 1963] 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 [Toropov and Alvarez 1998]. 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 [Vekeria and Parmee 1996, D. *et al.* 1998] 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 [El-Beltagy and Keane 1999] 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. Finally, a very recent approach [El-Beltagy *et al.* 1999] uses a functional approximation method to form reduced models. The approximation consists of a base-line polynomial and several Gaussian functions centered at some sample points to account for local deviations. The approach is very promising but suffers from some numerical difficulties especially when using relatively close sample points.

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 (unfortunately a detailed discussion of this is beyond the scope of this discussion). To the best of our knowledge, none of these approaches addressed the problem of unevaluable points.

In contrast, this proposal presents several approaches that suit a variety of conditions and situations. We present approaches ranging from ones 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, to ones in which the cheap model needs to be a good predictor of the accurate one. We take into consideration the typical characteristics of the engineering design optimization spaces such as the presence of numerous unevaluable points/regions. We address both situations in which a reduced model exists a priori and situations in which the reduced model needs to be learned online and for this case, we present several approach of the online learning of reduced models. Finally, we present concrete instantiations of our ideas, that have already been implemented and gave very promising results.

We begin this proposal with a description of the optimization environment that we are using in this work, including the genetic algorithm we will be using, and the application testbeds for the work. We then present the meat of the proposed work, our approaches for forming and using reduced models. We conclude with a discussion of related work, contrasting our work from others in this area. We include as part of this 15-page project description "appendices" containing the results of our initial exploratory work on this problem, which gives some idea of how we plan to evaluate our work, as well as the Co-PI's results from prior support.

2 Background

2.1 GADO: A Genetic Algorithm for Design Optimization

GADO [Rasheed 1998, Rasheed *et al.* 1997] is a highly adaptive 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 which typically arise in such applications. GADO has been applied in a variety of optimization tasks which span many fields. It demonstrated a great deal of robustness and efficiency relative to competing methods [Rasheed 1998].

In GADO, each individual in the GA population represents a parametric description of an artifact, such as an aircraft or a missile. All parameters have continuous intervals. 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 new crossover and mutation operators are used, which were designed specifically for the target domain type.

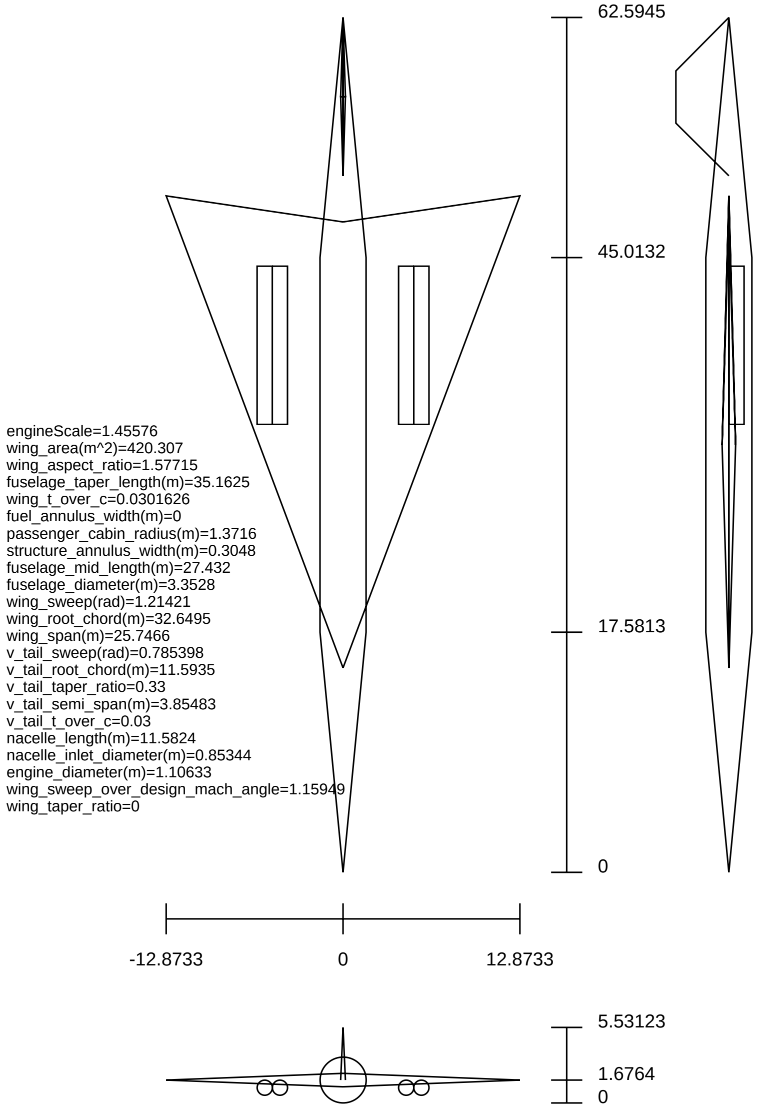
2.2 Test Domains

In the course of our earlier work we used two realistic engineering tasks to evaluate our efforts. We describe those two domains in this section, as well as additional domains to be used in this work.

2.2.1 Application Domain 1: Supersonic Transport Aircraft Design

Our first domain concerns the conceptual design of supersonic transport aircraft. We summarize it briefly here; it is described in more detail elsewhere [Gelsey *et al.* 1996]. 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 in Table 1 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.



```
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)

Table 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(r7)
5	engine size
6	wing area
7	wing aspect ratio
8	fuselage taper length
9	effective structural t/c
10	wing sweep over design mach angle
11	wing taper ratio
12	Fuel Annulus Width

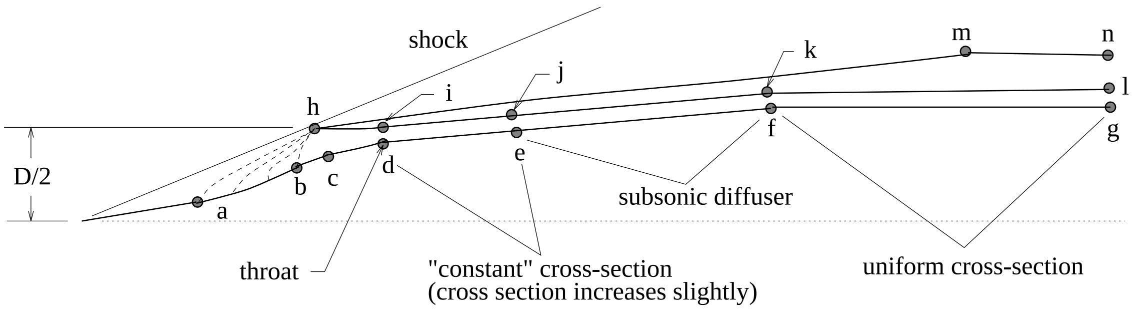


Figure 2: Supersonic missile inlet geometry model

Table 2: Inlet Parameters to Optimize

No.	*Parameter*	*Definition*
1	θ_1	initial cone angle
2	θ_2	final cone angle
3	x_d	axial location of throat
4	r_d	radial location of throat
5	x_e	axial location of end of "constant" cross section
6	θ_3	internal cowl lip angle
7	H_{ej}	height at end of constant cross section
8	H_{fk}	height at beginning of constant internal cross section

In summary, the problem has 12 parameters and 37 inequality constraints. 0.6% of the search space is evaluable.[1]

2.2.2 Application Domain 2: Supersonic Cruise Missile Inlets

Our second domain concerns the design of inlets for supersonic and hypersonic missiles. We summarize it briefly here; it is described in more detail in [Zha *et al.* 1996].

The missile inlet designed is an axisymmetric mixed compression inlet that cruises at Mach 4 at 60000 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 2 shows the model of the missile geometry which is composed of six fixed parameters and eight design parameters given in Table 2. The missile inlet is axisymmetric and the coordinates are given in terms of axial (x) and radial (r) positions.

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 [Haas *et al.* 1992]. It takes about 6 CPU seconds on a DEC Alpha 250 4/266 desktop workstation to evaluate each inlet.

The eight design parameters (all continuous valued) are given in Table 2, with coordinates given in terms of axial (x) and radial (r) positions. The goal of the optimization is to maximize the total pressure recovery,

[1]No statistics exist regarding the fraction of the search space that is feasible because it is extremely small.

a quantity that is commonly used to measure the performance of inlets.

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

2.2.3 New Inlet Design Domains and Simulators

As part of our intended collaboration with Aerospatiale-Matra (France), we intend to explore several inlet design tasks using simulators and analysis codes developed by Aerospatiale-Matra. These include:

The OCEAS Flow Solver OCEAS has been developed at Aerospatiale-Matra (France) to assist engineers in the aerodynamic design of missile inlets in a short period of time. It is a semi-empirical flow solver which uses simple but accurate physical models that require little CPU time. When the inlet is started, that is to say when the flow field is supersonic after the cowl entrance until a certain point within the inlet's duct, the supersonic and subsonic region of the flow field are separated by a vertical strong shock. OCEAS treats those two regions separately. The supersonic flow field is calculated first. Shocks are computed by solving the Rankine-Hugoniot equations. Expansion fans are modelized by one expansion wave calculated using the Prandtl Meyer formula and mass flow rate conservation. Boundary layers can also be modelized by ramp displacement; however, this artifact will not be used for this study. The subsonic flow field is not computed but losses within the diffuser are approximated by simple empirical formulae which take into consideration sudden expand, friction and separation at the throat. The maximum value of the aerodynamic coefficients are achieved when the strong shock is near the throat. Therefore, OCEAS computes several flow fields corresponding to different position of the strong shock near the throat : 12 computations, starting from 6 mm before the throat location, ending 6 mm after. Values of the total pressure recovery and mass flow rate are obtained from the characteristic curve of the inlet, which has a maximum called the critical point. Each computation only requires 6 CPU seconds using a DEC ALPHA 2100 work station. This computation time is far less than the time required for one Navier-Stokes computation.

The 2ES2D Flow Solver The accuracy and usability of any automated design process is mainly grounded on its performance estimation module. Indeed, the analysis part of the process must be accurate enough to predict the trend between candidate inlets and fast enough for the several hundreds of inlet analyses needed by the optimization to be performed within an acceptable time frame. To answer these requirements a hybrid flow solver, called 2ES2D (an acronym for Euler + Semi-Empirical Simulation), has been implemented by Aerospatiale-Matra, to be used in optimization.

The compression which occurs in the inlet must be achieved with the minimum total pressure loss. Moreover, the engine requires a minimum amount of flow to be captured by the inlet to work in optimal conditions. First, a supersonic compression is performed through a series of oblique shocks. The flow remains supersonic until the throat region, where a shock system close to a normal shock occurs, downstream of the geometrical throat. After this shock system, the flow is subsonic and is compressed along a diverging duct which acts as a subsonic diffuser. Basically, total pressure losses occur across the various shocks and through the viscous effects in the subsonic diffuser. The methodology developed for inlet aerodynamic performance evaluation is based on the different stages of the inlet. Thus, 2ES2D first uses a Euler simulation to account for the supersonic compression occuring above the inlet ramps. Then corrections are applied for the loss through the terminal shock system and for the viscous losses through the subsonic diffuser.

2.2.4 Other Domains

In addition to the above domains, we plan to use the following tasks as additional testbeds for this research.

Benchmark Engineering Design Domains In 1977, Eric Sandgren published his Ph.D. thesis by the title "The utility of nonlinear programming algorithms" [Sandgren 1977]. He applied 35 nonlinear optimization algorithms to 30 engineering design optimization problems and compared their performance. The 30 problems he used were also used by other researchers before and after Sandgren's work [Powell and Skolnick 1993]. We have used eight of these domains in our previous research and intend to use these eight, as well as others, in this research. More detailed description of these domains can be found in [Sandgren 1977, Powell and Skolnick 1993, Rasheed 1998].

3 Using Reduced Models to Speed Up Optimization

In our previous research we have encountered domains in which several models of the same artifact are available (for example different simulators of an inlet). If multiple models with different levels of abstraction exist — where the more accurate models are expected to be computationally more expensive — an intelligent method can be used to take advantage of all models. Several possibilities exist depending on the relative as well as the absolute complexities of the various models.

If the accurate model is extremely expensive, there is very little room for intelligence. (See [Zha *et al.* 1997] for our approach in such a case in the domain of missile inlet design, where the more expensive fitness function required about one week per simulation.) The case in which we are more interested is when the accurate model is not prohibitively expensive but still quite expensive relative to the reduced model (say the reduced model takes one second and the accurate one takes a minute). We intend to explore several ideas for using cheaper models to speed up the search without severely compromising quality.

More specifically we want to compare and contrast the following ideas and possibly others:

- Optimize only in the accurate model space and use the reduced model to provide guidance by, for example, doing some explorations to decide on which among several potential points is the most promising.
- Optimize only in the reduced model space and use the accurate model to ascertain the quality of promising points (say whenever a point is discovered that appears to be better than the best so far). A penalty may be incurred by the point and its neighborhood in the search space if its evaluation by the accurate model is significantly inferior to its evaluation with the crude model.
- Optimize in **both** model spaces simultaneously with the accurate model optimization being the **primary** focus and the crude model optimization acting as a source of *inspiration* in the following sense: Whenever the optimization in the reduced model space appears to have discovered a point that is significantly better than the best point found so far in the accurate-model optimization, the accurate-model optimizer is forced to evaluate and consider that promising point. The reduced-model optimization will proceed in a much faster pace than the primary accurate-model optimization and thus it will explore the search space more thoroughly and may well speedup the primary optimization depending on the crude model accuracy.

In the remainder of this section we describe a specific approach that we already implemented. It is intended to give an idea about our experimentation and evaluation methodologies. We still have to explore lots of different variations and possibilities but it is still a fairly well defined method.

3.1 A concrete example: Informed GA 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. 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 [Rasheed 1998] 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 [Rasheed 1999] 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 according to the method described in [Rasheed 1999]. 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.

The initial results of applying the informed operators, using pre-existing or online reduced models, to speed up the optimization in the aircraft design domain are included in the appendix below.

4 Formation of On-line Reduced Models

In domains in which no reduced models exist a priori, we could still take advantage of the techniques described in the previous section by forming the reduced models (and continuously reforming and improving them) on-line during the optimization. We intend to investigate and compare several approaches for doing this. The approaches we are planning to investigate can be divided into:

- **Global Approaches** which attempt to form a reduced model that is applicable in all parts of the search space. We need high order and/or piece-wise approximation functions to accomplish this. We are particularly interested in the use of spline-functions. We do not expect, however, to be able to rely solely on global approximations. We have seen design spaces with lots of discontinuities in the function and derivatives, high degrees of non-linearity and all sorts of numerical noises and problems. A more realistic expectation is to use the global approximation in conjunction with local (clustered) approximations to achieve a better quality than using only one type.

- **Local (Clustered) Approaches** which only approximate a limited part of the search space. By breaking the space into smaller parts, lower order approximation functions can be used to approximate each part. For these methods to work we need to sample the search space and then break the sample into clusters or subgroups. The clustering could be done in a variety of ways. In particular, we would like to explore two broad lines:

  – **Few Large-Clusters:** This approach maintains a relatively small number of clusters which contain many points each (hundreds or possibly thousands of points). The clustering overhead is less with this approach but its harder to fit each cluster and high order approximation functions may be necessary.

  - **Many Small-Clusters:** This approach maintains a much larger number of smaller clusters (typically in the order of ten points each). The increased overhead of clustering may be compensated for by using much simpler approximation functions (constants or linear approximations).

Our intuition is that the large-cluster approach is probably going to be the winner in most engineering design optimization spaces, because of the high non-linearity and dimensionality. Nevertheless, we feel that we should experimentally investigate the conditions under which either approach my be superior.

The online learned reduced models can be used in most of the approaches described in the previous section in any place a reduced model is to be used. Moreover, there is an added advantage to generating online reduced models. Since these models are algebraic, they can easily be explored and manipulated with a variety of techniques. For example, we could determine from the approximation function whether the problem is decomposable (i.e. can be broken into several subproblems). This will be evident if, for example, the cross terms of the form $a_{ij}x_ix_j, i \neq j$ in a quadratic approximation are much smaller than other terms. We can also determine which variables are more important than others. There is a wide range of possibilities that we intend to explore in this regard.

In the remainder of this section we describe a specific approach that we already implemented. It is intended to give an idea about our experimentation and evaluation methodologies. We still have to explore lots of different variations and possibilities but it is still a fairly well defined method.

4.1 A concrete example: Incremental-Approximate-Clustering Based Reduced Models

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 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. When the sample reaches its maximum size, new points replace existing points at random. As a result the sample will always contain more newly-visited points than old points which is desirable as more attention should go to the search space regions currently being exploited by the optimizer.

4.1.1 Incremental approximate clustering

We keep the sample divided into clusters. Starting with one cluster, we introduce one more cluster every specific number of iterations. This number of iterations (which becomes the average cluster size) should be small enough to allow reasonably correct approximation of a potentially complex function with quadratic approximations over the cluster points. The average cluster size should also be big enough to allow for least squares quadratic fitting which contains a quadratic number of coefficients. We introduce one new cluster at iteration intervals equal to four times the size of the GA population. 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. When a point joins a cluster, the cluster center is immediately recomputed to be the center of gravity of all the points in the cluster. In the case of a limited size sample, when a new point replaces and old point, the cluster losing a point is immediately updated. If a cluster lost its last point as a result of point replacement, the cluster will take and consist of the replacing point.

From the above discussion, it can be observed that we maintain the invariant that the cluster center of each cluster is the center of gravity of the points currently belonging to the cluster at all times. We do not maintain, however, that each point belongs to the cluster whose center is closest to the point at all times (this is why we call this approximate-clustering) because it could be very expensive to do that and in fact is not necessary. Instead, we "refresh" the clusters every time a new cluster is added. We do this by doing multiple passes on all the points in the sample reassigning every point to the right cluster. We repeat this process until in one pass we encounter no more "mis-clustered" points than 5% of the current sample size.

Again, in the case of a limited size sample, we stop adding clusters when the sample reaches its maximum size but we continue to refresh the clusters at the same intervals.

4.1.2 Formation of the approximation functions

We distinguish between the approximation functions for the measure of merit and those for the sum of constraints.[2] 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 [Press *et al.* 1992]. It works by using the normal equations method to minimize:

  $$\chi^2 = \sum_{k=1}^{N} \left[\frac{F(\bar{X_k}) - \hat{F}(\bar{X_k})}{\sigma_k} \right]^2$$

  where N is the number of evaluable points in the sample, $F(\bar{X_k})$ is the true value of the simulator output (measure of merit or violation sum) of point $\bar{X_k}$, and σ_k is the error in measurement (standard deviation) associated with point $\bar{X_k}$. We use these standard deviations in a novel way. We assign $\sigma_k = |F(\bar{X_k})|$ unless $|F(\bar{X_k})|$ is less than 10^{-3} in which case we assign $\sigma_k = 10^{-3}$ to prevent division problems. Thus, no small number of outliers will catastrophicly affect the fitting. In the case of fitting the sum of constraint violations, for feasible points, we assign $\sigma_k = 1000$ so that the approximation will not pay too much attention to forcing the values to be zero when in fact it is not important — approximating the sum of violations of a feasible point by a negative value is perfectly appropriate as we will force it to zero at the time of approximate evaluation. Moreover, since we use a two phase approach for approximate evaluation (described below), we do not use the approximation function for the violation at all if the point is classified as feasible.

- **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 points. We have found that clusters which have a number of evaluable points bigger than or equal to 1.5 times the number of constants in the quadratic approximation can be safely fit. Smaller clusters can lead to numerical problems in the curve fitting routine such as matrix singularities. We tried another method for least squares fitting based on single value decomposition but it did not do any better and was much slower. We also experimented with using linear approximation functions for clusters with a medium number of evaluable points but this did not improve the performance.

4.1.3 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:

[2]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.

- 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 five nearest neighbors of the new point to guess whether the point is likely to be feasible, infeasible-evaluable or unevaluable. We do this by computing three scores: the *feasible score*, the *infeasible-evaluable score* and the *unevaluable score*. The *feasible score* is the sum of the inverse distance functions from the point to all the *feasible* points among its five nearest neighbors. The other scores are computed in a similar manner. The inverse distance functions we used were the inverse of the exponential of the distance, to avoid division problems. If the point is classified as unevaluable, a very large fictitious fitness is returned. If the point is classified as feasible, zero is returned for its violation sum and the approximation is used to compute its measure of merit which is the same as its fitness. Otherwise, both the measure of merit and violation are computed using the approximations and, together with the current penalty coefficient, are used to compute the approximate fitness. We only use the two phase approach in the second half of the optimization because it relies on neighborhood information that could be misleading in the early stages of the optimization when the search space is only very sparsely covered with points. We have also verified experimentally that using the two phase approach in the second half of the optimization improves performance over just continuing to use the one phase approach.

5 Final Remarks

A genetic algorithm cannot be cavalier in its calls to its fitness function when the fitness function has high computational complexity, such as is the case in many engineering design optimization tasks. In this work we have proposed extending our previous work on this problem through the use and discovery of reduced models. The use of realistic tasks with expensive fitness drives our work towards the creation of methods that can be proven to work on a class of hard problems of real importance.

This proposal presents several approaches that suit a variety of conditions and situations. We take into consideration the typical characteristics of the engineering design optimization spaces such as the presence of numerous unevaluable points/regions. We address both situations in which a reduced model exists a priori and situations in which the reduced model needs to be learned online and for this case, we present several approach of the online learning of reduced models. Finally, we present concrete instantiations of our ideas, that have already been implemented and gave very promising results.

This research has a considerable potential impact on the design optimization methodology. We expect several high quality publications to result. We also plan to transfer the technology to our industrial collaborators which have always provided valuable feedback from real life applications and experiences. We expect to be able to considerably speed up design optimization. We also expect to gain more understanding of the conditions in which different methods are more appropriate than others, how to recognize these conditions, and how much improvement is expected under the different conditions.

Appendix I: Initial Results

Figure 3 demonstrates the utility of the reduced-model-based 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 more aggressive values relative to the defaults published in [Rasheed 1998] to speedup the experiments and show how the reduced-model-based 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

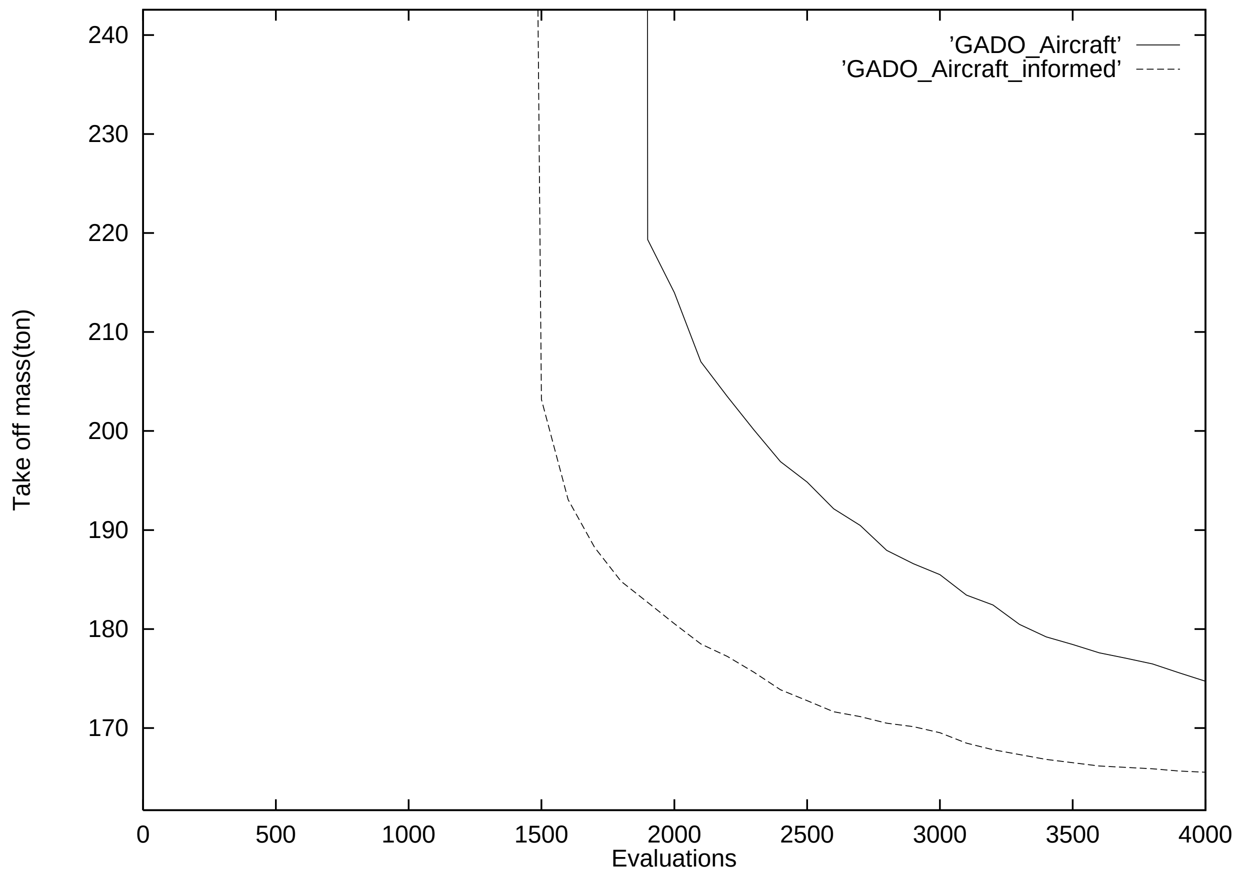


Figure 3: Effect of reduced-model-based informed operators on average performance in application domain 1 (aircraft design)

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 reduced-model-based informed operators improved the performance in all stages of the search. The feasible region was reached faster with the reduced-model-based informed operators.[3]

Figure 3 demonstrates how GADO with the reduced-model-based 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 reduced-model-based informed operators, provided enough time was given and more conservative settings were used (larger population for example). Figure 4 compares GADO with informed operators and aggressive settings given 4000 iterations to GADO with its original configuration used in [Rasheed 1998, Rasheed and Hirsh 1999] and given 12000 iterations. We find that GADO with the reduced-model-based 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

[3]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.

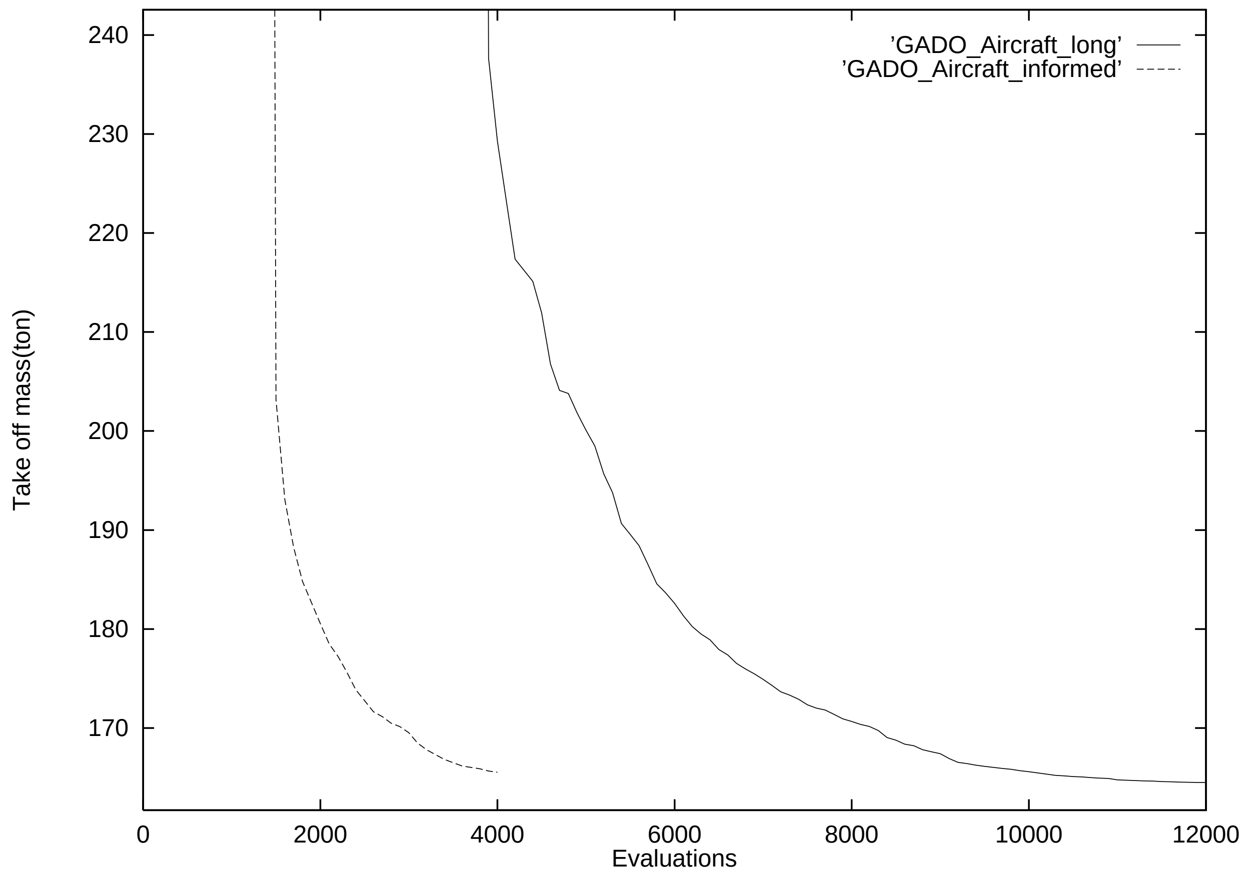


Figure 4: Speed-up comparison in application domain 1 (aircraft design). Note the horizontal scale difference from the previous figure.

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 5. The figure again demonstrates how a reduced model, even if only approximate, can be used with informed operators to great advantage.

Appendix II: Results from Prior NSF Support

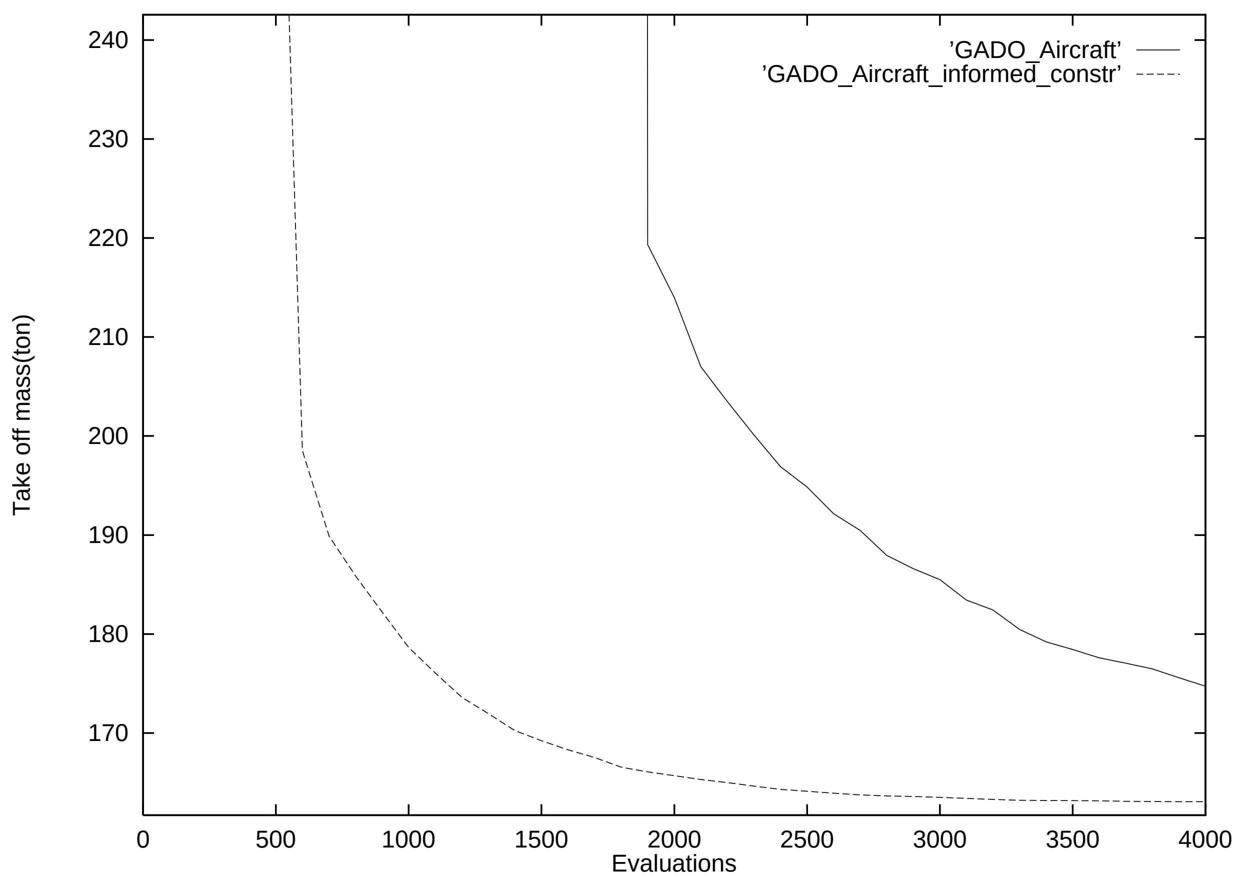


Figure 5: Effect of constraint based informed operators on average performance in application domain 1 (aircraft design)

References

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

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

[El-Beltagy and Keane 1999] 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.

[El-Beltagy *et al.* 1999] 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.

[Gelsey *et al.* 1996] 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.

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

[Powell and Skolnick 1993] 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.

[Press *et al.* 1992] 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.

[Rasheed and Hirsh 1999] 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.

[Rasheed *et al.* 1997] Khaled Rasheed, Haym Hirsh, and Andrew Gelsey. A genetic algorithm for continuous design space search. *Artificial Intelligence in Engineering*, 11(3):295–305, 1997. Elsevier Science Ltd.

[Rasheed 1998] 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.

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

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

[Toropov and Alvarez 1998] 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.

[Vekeria and Parmee 1996] 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.

[Zha *et al.* 1996] 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.

[Zha *et al.* 1997] G.-C. Zha, D. Smith, M. Schwabacher, K. Rasheed, A. Gelsey, D. Knight, and M. Haas. High performance supersonic missile inlet design using automated optimization. *AIAA Journal of Aircraft*, 34(6), 1997.

Biographical Sketch

Haym Hirsh
Computer Science Department
Rutgers University
Piscataway, NJ 08855
732–445–4176
hirsh@cs.rutgers.edu
http://www.cs.rutgers.edu/~hirsh

Education

- Ph.D. in Computer Science, Stanford University, 1989
- M.S. in Computer Science: Artificial Intelligence, Stanford University, 1985
- B.S. in Mathematics/Computer Science, UCLA, 1983

Employment History

- July 1995–present:
  Associate Professor, Department of Computer Science, Rutgers University
- November 1997–December 1997:
  Consultant, AT&T Laboratories
- September 1997–December 1997:
  Visiting Associate Professor, Department of Electrical Engineering and Computer Science, MIT
  Visiting Scholar, Artificial Intelligence Laboratory, MIT
  Visiting Scholar, Laboratory for Computer Science, MIT
- September 1995–December 1995:
  Visiting Professor, Department of Computer Science, Carnegie Mellon University
- September 1989–June 1995:
  Assistant Professor, Department of Computer Science, Rutgers University
- April 1994–June 1995:
  Consultant, Bellcore
- May 1991–December 1993:
  Consultant, AT&T Bell Laboratories

Brief Biography

Professor Hirsh's recent research has concerned scientific and engineering applications of machine-learning and information retrieval methods, especially to tasks involving engineering design, bioinformatics, and human-computer interaction. His paper with Michiel Noordewier at the Tenth IEEE Conference on Artificial Intelligence for Applications in 1994 received the conference's Best Paper award. He was co-chair of the Eleventh International Conference on Machine Learning (with William Cohen, AT&T Bell Laboratories) in 1994, and co-chair (with Marti Hearst, Xerox PARC) of the 1995 AAAI Spring Symposium on Machine Learning in Information Access. He an associate editor of the *Journal of Artificial Intelligence Research*.

Relevant Publications

- Mark Schwabacher, Tom Ellman, and Haym Hirsh. Learning to set up numerical optimizations of engineering designs. *Artificial Intelligence in Engineering Design and Manufacturing*, 12(2), 1998.
- Khaled Rasheed, Haym Hirsh, and Andrew Gelsey. A Genetic Algorithm for Continuous Design Space Search. *Artificial Intelligence in Engineering*, 11(3):295–305, 1997.
- Khaled Rasheed and Haym Hirsh. Using Case Based Learning to Improve Genetic Algorithm Based Design Optimization. In *Proceedings of the Seventh International Conference on Genetic Algorithms* (ICGA'97), July, 1997.

Other Recent Publications

- Haym Hirsh and Michiel Noordewier. Using Background Knowledge to Improve Inductive Learning: A Case Study in Molecular Biology. *IEEE Expert*, 9(5):3–6, October, 1994.
- Ronen Feldman, Ido Dagan, and Haym Hirsh. Mining Text Using Keyword Distributions, *Intelligent Information Systems.* To appear, 1998.
- Ronen Feldman and Haym Hirsh. Exploiting Background Information in Knowledge Discovery from Text. *Intelligent Information Systems.* 9:1 83-97, July 1997.

Collaborations within the Last 48 Months: Beth Adelson (Rutgers), Arunava Banerjee (Rutgers), Chumki Basu (Rutgers, Bellcore), Helen Berman (Rutgers), William Cohen (AT&T), Ido Dagan (Bar-Ilan), Brian Davison (Rutgers), Tom Ellman (Rutgers), Ronen Feldman (Bar-Ilan), Andrew Gelsey (Rutgers, Goldman Sachs), David Goodman (Rutgers), Nathalie Japkowicz (Rutgers), Paul Kantor (Rutgers), Daniel Kudenko (Rutgers), David Loewenstern (Rutgers, Lucent, NEC), Nina Mishra (University of Illinois), Michiel Noordewier (Rutgers, Smith Kline Beecham), Leonard Pitt (University of Illinois), Khaled Rasheed (Rutgers), Gerard Richter (Rutgers), Mark Schwabacher (Rutgers, NIST), Gary Weiss (Rutgers, AT&T), Peter Yianilos (NEC).

Graduate Students Advised: Vipul Kashyap (MCC), David Lubinsky (University of Witwatersrand), Steven Norton (AT&T), Lorien Pratt (Evolving Systems), Khaled Rasheed (Rutgers)

PhD Advisor: Bruce Buchanan

Khaled Rasheed
Computer Science Department
Rutgers University
Piscataway, NJ 08855
732–445–4379
krasheed@cs.rutgers.edu
http://www.cs.rutgers.edu/~krasheed

Education

- Ph.D. in Computer Science, Rutgers University, 1998
- M.S. in Computer Science, Rutgers University, 1995
- B.S. in Computer Science and Automatic Control, Alexandria University, Egypt, 1990

Employment History

- July 1999–present:
  Assistant Research Professor, Department of Computer Science, Rutgers University
- January 1998–June 1999:
  Research Associate, Department of Computer Science, Rutgers University
- June 1994–December 1997:
  Research Assistant, Department of Computer Science, Rutgers University
- August 1993–May 1994:
  Teaching Assistant, Department of Computer Science, Rutgers University
- August 1992–May 1993:
  Teaching Assistant, Department of Computer Science, Iowa State University
- October 1990 – August 1992:
  Teaching Assistant, Department of Computer Science and Automatic Control, Alexandria University, Egypt

Brief Biography

Dr. Khaled Rasheed is a Research Associate at the Computer Science department, Rutgers University - New Brunswick. He received his Ph.D. from Rutgers University in January 1998 on his work on adapting genetic algorithms for problems in engineering design. His research interests include artificial intelligence, genetic algorithms, design optimization, bioinformatics, and machine learning.

Relevant Publications

- Khaled Rasheed. *GADO: A Genetic Algorithm for Continuous Design Optimization.* Ph.D. Thesis, Computer Science Department, Rutgers University, 1998
- Khaled Rasheed, Haym Hirsh, and Andrew Gelsey. A Genetic Algorithm for Continuous Design Space Search. *Artificial Intelligence in Engineering*, 11(3):295–305, 1997.
- G.-C. Zha, D. Smith, M. Schwabacher, K. Rasheed, A. Gelsey, D.Knight and Martin Haas. High Performance Supersonic Missile Inlet Design Using Automated Optimization. *Journal of Aircraft*, 34(6), 1997.

- Khaled Rasheed and Haym Hirsh. Using Case Based Learning to Improve Genetic Algorithm Based Design Optimization. In *Proceedings of the Seventh International Conference on Genetic Algorithms* (ICGA'97), July, 1997.

- Michael Blaize, Doyle Knight, and Khaled Rasheed. Automated Optimal Design of Two dimensional High Speed Missile Inlets. The 36th AIAA Aerospace Sciences Meeting, 1998.

Other Significant Publications

- A. Gelsey, D. Smith, M. Schwabacher,K. Rasheed, and K. Miyake. "A Search Space Toolkit". Decision Support Systems, 18:341-356, 1996.

Collaborations within the Last 48 Months: Andrew Gelsey (Rutgers, Goldman Sachs), Michiel Noordewier (Rutgers, Smith Kline Beecham), Mark Schwabacher (Rutgers, NIST, NASA), Tom Ellman (Rutgers), Steve Garofalini (Rutgers), William Sofer (Rutgers), Peter Robinson (NASA), Mike Lowery (NASA).

PhD Advisor: Haym Hirsh

Budget Justification

The project support includes one third of the academic year salary of Dr. Khaled Rasheed, who holds the title of Assistant Research Professor, in each of the three years. The project support also includes one month summer salary for Dr. Haym Hirsh (Co-PI) in each of the three years. The figures reflect an estimated increase of 3%/year, based on information obtained from the Rutgers Faculty of Arts and Sciences Office of the Dean.

Also included is support for one Graduate Student — both academic and summer support. Based on a contractual agreement, graduate assistants work up to 15 hours/week during the academic year. The summer support is for two months at 30 hours/week. The summer salary reflects the increase in hours/week.

Fringe benefits are negotiated between Rutgers and the DHHS. In this case, the rates are effective July 1, 1998. There is no fringe benefit charge for faculty summer salary. GA-AY support includes a fringe benefit rate of 18.25%; GA summer support includes a fringe benefit rate of 7.5%. The indirect cost rate, also effective July 1, 1998, has been negotiated at 56%.

The budget includes equipment costs with a larger outlay for computing equipment and software for the first year, with more modest costs for updates to equipment, such as additional disk space, in subsequent years. All three years include equipment maintenance. This maintenance covers both hardware and software.

The budget also includes travel support for both the PIs and the graduate assistant at an annual budget of $2000/PI and $1000/GA.

Materials and supplies include estimates for basic equipment needs such as toner cartridges, paper, cables, books, etc

Publication costs include estimates for photocopying costs, page overrun costs, journal subscriptions, etc.

Also budgeted is tuition remission for the graduate assistants. Based on historical precedence, years 02 and 03 include a 5% increase.

Facilities, Equipment and Other Resources

As part of the general computing support provided by the Laboratory for Computer Science Research (LCSR) Computing Facility to all funded projects, the following environment and software is available for this proposed project's use. Additionally, our researchers use specialized equipment at other sites on the Internet.

The equipment and software falls into the following major categories:

- Individual workstations (primarily Sun Microsystems);
- Shared, multi-user computer systems (primarily Sun Microsystems);
- IBM SP-2 and NCUBE-2 parallel multiprocessor system;
- Workstation cluster (6 Sun Ultrasparcs) connected by a Myricom gigabit crossbar switch;
- Other shared facilities such as various printers and modems;
- Software languages (including Java, Prolog, Lisp, Eiffel, C, C++, and Fortran);
- Document preparation tools (such as TeX and Publisher); and
- Various specialized tools for specific research areas (such as Matlab and Maple).

Currently all faculty and graduate student offices have workstations equivalent in power to a Sparcstation 5. For compute-intensive work, a 3-processor, 1 GB system is available, with Digital's Alpha processors. All systems are networked. The network is based on a pair of Cisco 5000 switches, with 1 Gbps backplanes. Connections to most offices use 10 Mbps Ethernet. We are in the process of moving to switched 100 Mbps Ethernet for servers and other systems that need high-speed connections.

The LCSR Computing Facility is comprised of a total full-time staff of eight members. These staff support LCSR research, as well as instructional computing for the computer science department, and various projects for the University as a whole. The staff who most directly support research computing include:

- 2 hardware staff, doing installation and maintenance (along with additional part-time technicians);
- 3 Unix support staff, plus student programmers;
- 1 Macintosh support person;
- 1 full-time operator plus student operations staff; and
- an Associate Director.

These personnel provide a stable computing base available to all projects associated with the Laboratory for Computer Science Research.