Using Modeling Knowledge to Guide Design Space Search

Andrew Gelsey
gelsey@cs.rutgers.edu

Mark Schwabacher
schwabac@cs.rutgers.edu

Don Smith
dsmith@cs.rutgers.edu

Computer Science Department
Rutgers University
New Brunswick, NJ 08903
USA
(908) 445–2001, FAX –5691

Abstract

Automated search of a space of candidate designs seems an attractive way to improve the traditional engineering design process. To make this approach work, however, the automated design system must include both knowledge of the modeling limitations of the method used to evaluate candidate designs and also an effective way to use this knowledge to influence the search process. We suggest that a productive approach is to include this knowledge by implementing a set of *model constraint* functions which measure how much each modeling assumptions is violated, and to influence the search by using the values of these model constraint functions as constraint inputs to a standard constrained nonlinear optimization numerical method. We test this idea in the domain of conceptual design of supersonic transport aircraft, and our experiments indicate that our model constraint communication strategy can decrease the cost of design space search by **one or more orders of magnitude**.

To appear: Artificial Intelligence in Design '96, Stanford, CA, June 1996

ftp://www.cs.rutgers.edu/pub/technical-reports/hpcd-tr-34.ps.Z

1 Introduction

Automated search of a space of candidate designs seems an attractive way to improve the traditional engineering design process. Each step of such automated search requires evaluating the quality of candidate designs, and for complex artifacts (e.g., aircraft, our main example), this evaluation must be done by computational simulation. However, computational simulation is based on a model of the physics of the artifact, and this model will generally make simplifying assumptions in order to be computationally tractable. Most existing computational simulators are intended to be used by human experts, and thus they typically include no explicit representation of their modeling assumptions. Instead, it is assumed that the experts know enough to stay away from portions of the design space that will violate the simulator's assumptions.

For example, a typical assumption for an aircraft simulator might be that the wings won't stall. Stall is a physical phenomenon that occurs when a wing is operated at too high an angle of attack and therefore ceases to generate lift. The physics of stall is understood, and there is in principle no reason not to model it in a simulator. However, a human expert aircraft designer doesn't want to design a plane that stalls during normal operation, so he doesn't need a detailed prediction of stall behavior. The designer is satisfied with an incomplete model as long as he can recognize "impossibly high" lift coefficients and realize that the design he is considering would actually stall and thus should be discarded.

However, if the simulator is invoked by another program such as an automated search procedure rather than by a human expert, it is quite likely that in exploring the design space, the automated search procedure will examine designs which violate the simulator's assumptions, and for those candidate designs, the evaluation of the design quality computed by the simulator may be meaningless. Furthermore, this meaningless value may appear better than the value for any physically realizable design, thus leading the search procedure to a worthless but apparently very good design.

In our earlier work [Gelsey 1995b], we have investigated the types of modeling knowledge that are needed so that a simulator can be reliably invoked by another program, and we have described algorithms for detecting assumption violations and other problems that might lead to low-quality or unreliable simulation results. In the present paper, we address the question of how information about model assumption violations can be effectively communicated to an automated search procedure so that the search procedure can find candidate designs that don't violate model assumptions.

2 Communication Strategies

Strategies for communicating information about model violations to the search procedure include:

The Null Strategy: ignore the model violation — the search procedure uses whatever value happens to be computed by the inapplicable model for the quality of the candidate design.

The Boolean Strategy: when any model violation occurs, always give the search procedure a standard "very bad value" as the quality of the candidate design.

Model Constraints: when a candidate design is evaluated, give the search procedure not only a value for the quality of the candidate design, but also values for a set of "model constraint" functions which measure how much the various modeling assumptions are satisfied or violated.

Model Penalties: same as the model constraints strategy, except that only the value for the quality of the candidate design is returned to the search procedure, and that value is penalized in proportion to the amount by which the various modeling assumptions are violated.

In this paper we will focus primarily on the boolean strategy and model constraints. The null strategy is unlikely to be useful unless it coincidentally happens to be the same as either the boolean strategy or the model penalties strategy. The boolean strategy can be useful — its advantages include:

- easy to implement: as soon as a violation is detected, just return immediately with a standard "very bad" value for the objective function
- it can be used even with unconstrained search methods

The model constraints strategy is more complicated to implement than the boolean strategy, but our experimental results later in this paper show that when used with a search method that allows constraints, the performance of the model constraints strategy is considerably better than that of the boolean strategy. We don't investigate the model penalties strategy in this paper, but discuss possible uses for it in our Future Work section.

3 Aircraft Design

We have pursued our investigation in the domain of conceptual design of supersonic transport aircraft. Figure 1 shows a diagram of a typical airplane automatically designed by our software system to fly the mission shown in Figure 2, and Figure 3 shows a block diagram of the system's software architecture. The search controller attempts to find a good aircraft conceptual design for a particular mission by varying major aircraft parameters such as wing area, aspect ratio, engine size, etc. using a numerical optimization algorithm. The search controller evaluates candidate designs using a multidisciplinary simulator with which it communicates via the Model/Simulation Associate (MSA), which implements the various communication strategies described in the previous section. In our current implementation, the search controller'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. The simulator computes the takeoff mass of a particular aircraft design for a particular mission as follows:

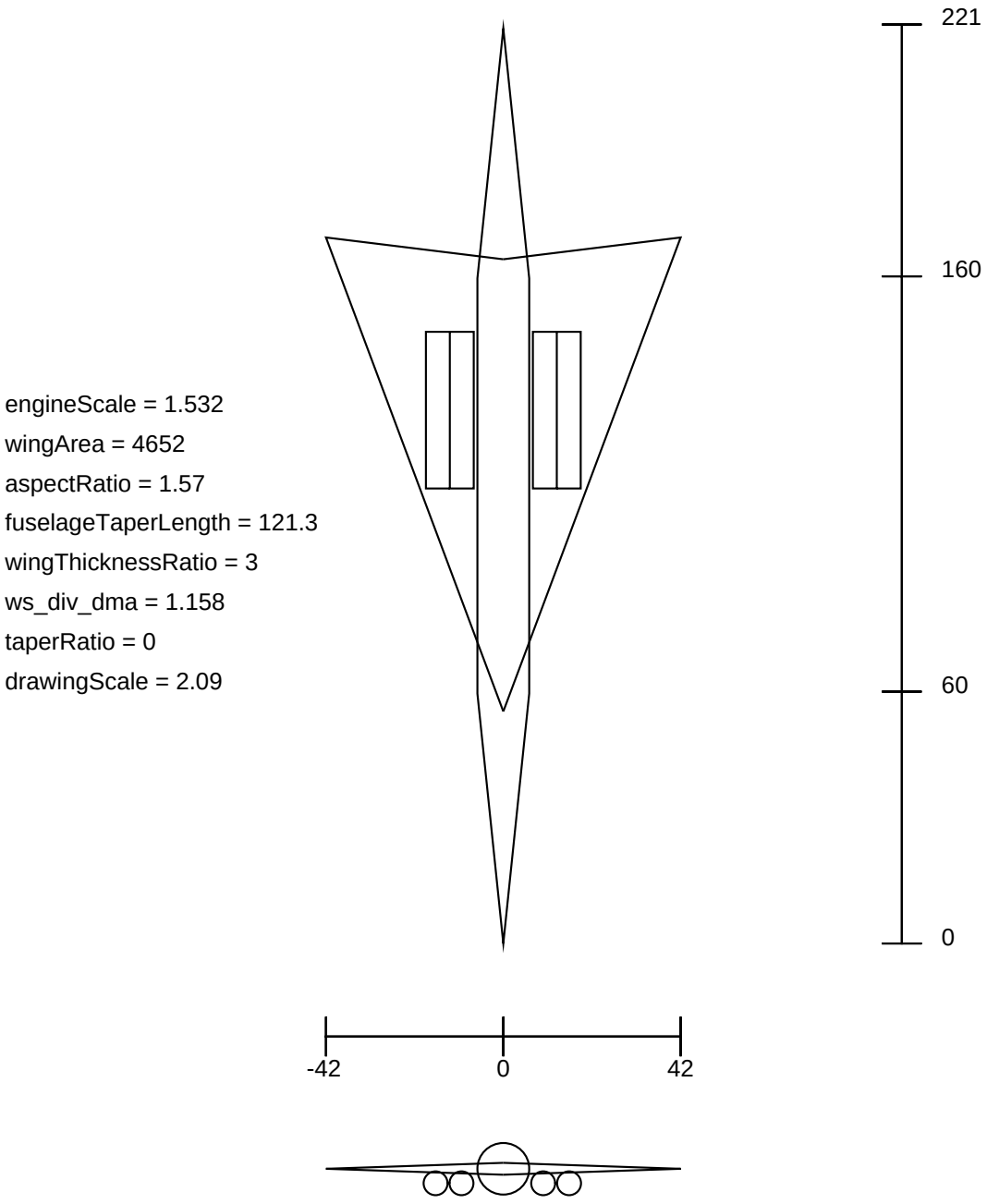

Figure 1: Supersonic transport aircraft designed by our system (dimensions in feet)

Phase	Mach	Altitude (ft.)	Duration (min.s)	comment
1	0.227	0	5	"takeoff"
2	0.85	40,000	85	subsonic cruise (over land)
3	2.0	60,000	180	supersonic cruise (over ocean)

capacity: 70 passengers.

Figure 2: Mission specification for aircraft in Figure 1

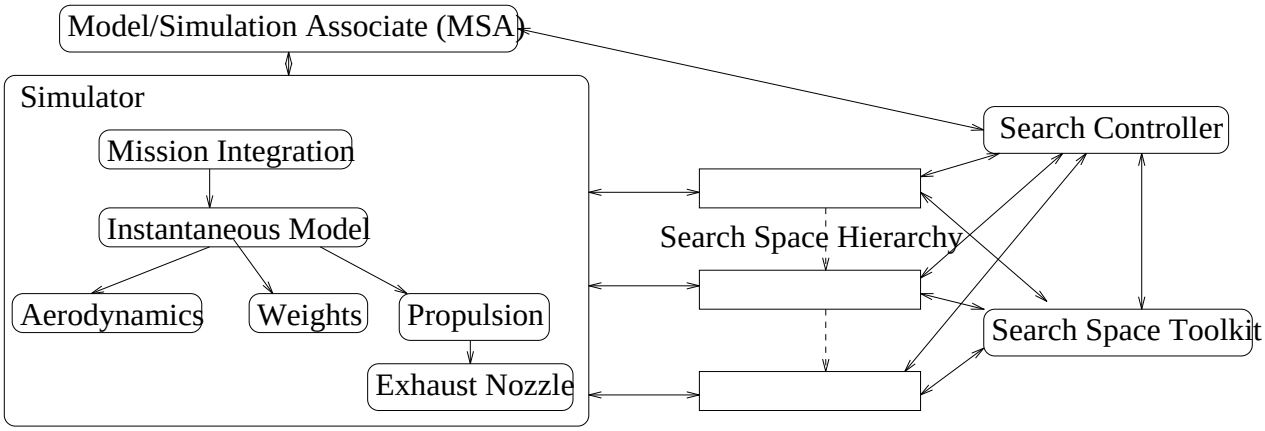

Figure 3: Software architecture block diagram

1. Compute "dry" mass using historical data to estimate the weight of the aircraft as a function of the design parameters and passenger capacity required for the mission.

2. Compute the landing mass $m(t_{\text{final}})$ which is the sum of the fuel reserve plus the "dry" mass.

3. Compute the takeoff mass by numerically solving the ordinary differential equation

$$\frac{dm}{dt} = f(m, t)$$

   which indicates that the rate at which the mass of the aircraft changes is equal to the rate of fuel consumption, which in turn is a function of the current mass of the aircraft and the current time in the mission. At each time step, the simulator's aerodynamic model is used to compute the current drag, and the simulator's propulsion model is used to compute the fuel consumption required to generate the thrust which will compensate for the current drag.

The software architecture in Figure 3 also includes a "search space toolkit" for determining the design space structure, which is described in [Gelsey and Smith 1995, Gelsey *et al.* 1996] and therefore will not be discussed further in this paper.

A complete mission simulation requires about 1/4 second of CPU time on a DEC Alpha 250 4/266 desktop workstation.

4 Search Procedure

In this paper we will focus on search of a space of candidate designs using numerical optimization methods which vary a set of continuous parameters to minimize[1] a nonlinear objective function subject to a set of nonlinear equality and inequality constraints. The numerical optimizer used in this paper is CFSQP [Lawrence *et al.* 1995], a state-of-the-art implementation of the Sequential Quadratic Programming method. Sequential Quadratic Programming is a quasi-Newton method that solves a nonlinear constrained optimization problem by fitting a sequence of quadratic programs[2] to it, and then solving each of these problems using a quadratic programming method.

In order to handle unevaluable points (i.e., points whose objective function was assigned the boolean strategy standard "very bad" value), we have supplemented CFSQP with *knowledge-based gradients*. Knowledge-based gradients are computed by using a set of rules that specify how to compute gradients with reasonable accuracy in the presence of unevaluable points. In addition, we have arranged for the line searches in CFSQP to terminate when they encounter unevaluable points. These enhancements to CFSQP are crucial when using the boolean communication strategy, which results in numerous unevaluable points. They can also be helpful when using the model constraints communication strategy, since some limitations of the simulator are not modeled in the model constraints, so some

[1] To instead *maximize* the objective function, just multiply it by -1 and minimize.

[2] A quadratic program consists of a quadratic objective function to be optimized, and a set of linear constraints.

unevaluable points exist even when using model constraints. In the experiments reported in this paper, with the boolean strategy, 76% of the points encountered were unevaluable, and with the model constraints strategy, 4% of the points encountered were unevaluable. (Note: the optimizer tends to avoid unevaluable points, so these percentages are considerably lower than the average density of unevaluable points in the search spaces, as indicated by the data presented later in this paper.) Knowledge-based gradients are further described in [Schwabacher and Gelsey 1996].

5 Model Constraints

For the experiments in this paper, the MSA module in Figure 3 computes the following model constraint functions, which are ≤ 0 if a constraint is satisfied and positive otherwise:

ETUB = <maximum throttle required during mission simulation> − <maximum throttle setting allowed for engine>. If an impossibly high throttle is required to fly the mission, the simulation will continue using extrapolation, but the value of ETUB will indicate the extent to which the engine model assumptions are violated.

ETLB = <minimum throttle setting allowed for engine> − <minimum throttle required during mission simulation>.

A1LB, A1UB, A2LB, A2UB: Similar to above — violation of bounds for a two-dimensional table of experimental data on supersonic drag.

WLUB = <maximum wing loading during mission simulation> − <maximum wing loading simulator can validly model>.

FM = <fuel mass that current candidate design requires to complete mission> − <fuel mass that can be stored in available volume for current candidate design>.

STALL = <maximum lift coefficient during mission simulation> − <maximum lift coefficient simulator can validly model>. The simulator assumes wings won't stall, and this constraint function computes how well that assumption is satisfied.

These model constraint functions are continuous and usually smooth with respect to the design parameters as their values change sign, which is very important so that when MSA is using the model constraint communication strategy, CFSQP (the numerical optimizer) can follow constraint boundaries if necessary as it searches for an aircraft design which can fly the given mission with minimal takeoff mass. If MSA is following the "boolean" communication strategy, it does not give the values of the model constraint functions to CFSQP: instead, any candidate design for which some model constraint function is positive will be evaluated to have a standard "very large" takeoff mass.

In addition to these model constraints, MSA computes the following design constraint:

PASS = <passenger capacity required for the mission> − <passenger capacity available with current design parameters>.

Note: differences between model constraints and design constraints include:

- Design constraints can be extracted directly from design goals, while formulating model constraints requires carefully examining the underlying assumptions of the model which the simulator is based on.

- Design constraints can be violated without reducing the quality (i.e., correctness) of the objective function computed by the simulator, but when a model constraint is violated, the value of the objective function computed by the simulator cannot be trusted. For example, even if the PASS constraint is violated, the simulator can still correctly compute the takeoff mass needed to fly though the mission carrying whatever number of passengers the aircraft is actually able to hold. However, if a model constraint is violated, then the takeoff mass computed by the simulator may be wildly wrong. For example, if the simulator is allowed to violate the STALL constraint, the optimizer may design an aircraft with very small wings operated at a very high angle of attack which may appear to be a very efficient aircraft, much better than the best physically plausible design, but which in fact is not capable of flying at all.

- If a design constraint happens to be inactive at the optimal design (i.e., the constraint is satisfied for all designs near the optimal design, so the optimum does not lie on a constraint boundary), then the "null" communication strategy will be effective when applied to this constraint — i.e., the constraint may safely be ignored without a detrimental effect on the optimization. However, the null communication strategy will not in general be effective when applied to model constraints, even if they are inactive at the optimal design. In the region where a model constraint is violated, the value of the objective function computed by the simulator may include random meaningless values, so therefore if the model constraint violations are ignored by the null strategy, the region where the model constraint is violated may include local optima of the objective function or, even worse, points having (spurious) values of the objective function better than the best value for any design satisfying all the model assumptions. Either of these conditions can "trap" the optimizer and keep if from getting to the true optimum, even though the model constraint in question is inactive at the true optimum.

6 Experimental Results

To experimentally test MSA communication strategies, we used a design space in which the optimizer varied the following aircraft conceptual design parameters over a continuous range of values:

1. engine size
2. wing area
3. wing aspect ratio
4. fuselage taper length (how "pointed" the fuselage is)

	Small Box		Big Box	
Design Parameter	low	high	low	high
engine size	0.5	3	0.1	5
wing area (sq. ft.)	1500	13500	500	20000
wing aspect ratio	1	2	0.5	3
fuselage taper length (ft.)	100	200	50	300
effective structural thickness over chord	1	5	0.5	10
wing sweep over design mach angle	1	1.45	0	1.45
wing taper ratio	0	0.1	0	0.1
fuel annulus width (ft.)	0	4	0	8

Figure 4: Subsets of design space explored

5. effective structural thickness over chord (a nondimensionalized measure of wing thickness)

6. wing sweep over design mach angle (a nondimensionalized measure of wing sweep)

7. wing taper ratio (wing tip chord divided by wing root chord)

8. fuel annulus width (space available in fuselage for fuel storage)

Figure 4 shows the two subsets we explored in the design space defined by these design parameters.

To test the effect of the MSA communication strategy on the design process, we considered the following strategy combinations:

1. Return values of all model constraint functions to the optimizer as nonlinear inequality constraints.

2. Return values of all model constraint functions except ETLB and ETUB to the optimizer, but for candidate designs where the engine table constraints were violated (ETLB or ETUB positive), use the "boolean" strategy and return a standard "very large" value for takeoff mass.

3. Return values of all model constraint functions except A1LB, A1UB, A2LB, and A2UB to the optimizer; use "boolean" strategy for points which required extrapolation outside the aerodynamics table bounds.

4. Return values of all model constraint functions except FM, which is "boolean".

5. Return values of all model constraint functions except STALL, which is "boolean".

6. Return values of all model constraint functions except WLUB, which is "boolean".

7. Use the "boolean" communication strategy for all model constraint functions.

Design Parameters:	
engine size	1.532
wing area	4652 sq. ft.
wing aspect ratio	1.570
fuselage taper length	121.3 ft.
effective structural thickness over chord	3.002
wing sweep over design mach angle	1.158
wing taper ratio	0
fuel annulus width	0
Objective Function:	
Takeoff Mass	167.4 tonnes
Model Constraints:	
ETUB	-41.57
ETLB	-0.76
A1LB	-2.2
A1UB	-1.8
A2LB	-1.5
A2UB	-8.5
WLUB	-149.8
FM	-0.0011 tonnes
STALL	0
Design Constraint:	
PASS	-2

Figure 5: Best design found for mission of Figure 2

8. A two-level approach in which the "boolean" communication strategy is used to find a feasible point, and then all model constraints are used to find an optimum.

For each strategy combination, our system randomly choose points in the "small box" until it found 74 "evaluable" points (i.e., points whose objective function was not assigned the boolean strategy standard "very bad" value).[3] Each of these 74 points was then used as a starting point for a design optimization using CFSQP to try to find an optimal aircraft design for the mission shown in Figure 2. (We required the starting points to be evaluable because if CFSQP happened to be started in an unevaluable region, then all components of the gradient would be zero and the optimization would terminate immediately.) The best design found for this mission in all the experiments is shown in Figure 5, and a diagram of this aircraft appears in Figure 1.

[3] 74 is not a "magic" number; it was just a convenient choice given available disk space.

The performance of the strategy combinations is shown in a table in Figure 6. The "Success" column for each strategy combination shows what fraction of the 74 optimizations found aircraft designs having takeoff masses within 1% of the takeoff mass of the apparent "global optimum" — the best design we found for this mission (Figure 5). The "Start Cost" column shows how many simulations had to be run on unevaluable points while finding the 74 optimization starting points, and "Opt. Cost" shows the total number of simulations that were run during each set of 74 optimizations.[4] The "Est. 99% Cost" column in Figure 6 gives the estimated cost with each strategy combination to have a 99% chance of finding the global optimum, which is computed by multiplying the average cost per optimization times $\log(1-P_{\text{desired}})/\log(1-P_{\text{success}})$, where P_{desired} is the desired probability of finding the global optimum (99% in this case) and P_{success} is the probability of any single optimization finding the global optimum (which we estimate with the value in the "Success" column). Figure 7 shows graphically the "Est. 99% Cost" to achieve a range of different design qualities. (Note that the curves for some of the strategies are so bad that they are above the largest vertical axis value shown and therefore they do not appear in the plot. See the "Est. 99% Cost" column in Figure 6 for their values at Quality = 1.01.)

The data in Figure 6 indicates that the model constraints communication strategy can find the global optimum with a 99% confidence at a cost which is **one or more orders of magnitude smaller** than the cost to achieve comparable results with the boolean communication strategy. Examination of the different strategy combinations indicates that the model constraints which contribute most to this performance difference are the constraints active at the global optimum (constraint values ≈ 0 in Figure 5), but that even the constraints which are inactive at the global optimum may give a factor of two to three speedup when handled using model constraints rather than the boolean strategy. If a constraint is active at the global optimum, then CFSQP must "navigate" along the constraint boundary when searching for the optimum. This navigation is easy when the boundary is defined by a smooth model constraint, but much more difficult when the boundary is marked only by a sudden jump in the objective function from a reasonable value to the boolean "very bad" value. Model constraints which are inactive at the optimum may still be active during some parts of the search and thus can help guide the search and prevent the optimizer from getting stuck.

Model constraints which are active at the global optimum are more critical, but it is important to note that there will typically be no reliable *a priori* way to determine which model constraints will be active at the global optimum. This fact suggests that the model constraints communication strategy should be used to handle all model assumptions, even though implementing smooth model constraint functions may require more work than implementing the simpler boolean communication strategy.

An issue that should be considered is the question of why any model constraints are active for the globally optimal design. Does this situation indicate that there are actually better designs on the other side of the constraint boundary which the optimizer would be able to find if only we had a more sophisticated model that didn't need as many constraints? Not necessarily. For example, lift initially rises as a function of angle of attack and later

[4] As mentioned earlier, a complete mission simulation requires about 1/4 second of CPU time on a DEC Alpha 250 4/266 workstation.

Strategy Combination	Success	Start Cost	Opt. Cost	Est. 99% Cost
All model constraints returned	65/74	16	42375	1252
ETLB and ETUB "boolean"	52/74	3203	67158	3609
FM "boolean"	0/74	603	99215	$\gg$ 456565
STALL "boolean"	18/74	5441	81566	19427
A1LB/A1UB/A2LB/A2UB "boolean"	67/74	57	47042	1242
WLUB "boolean"	62/74	721	39404	1372
All model constraints "boolean"	0/74	21946	75804	$\gg$ 447106
Two level	72/74	18098	39389	990

Figure 6: Performance of the various strategy combinations

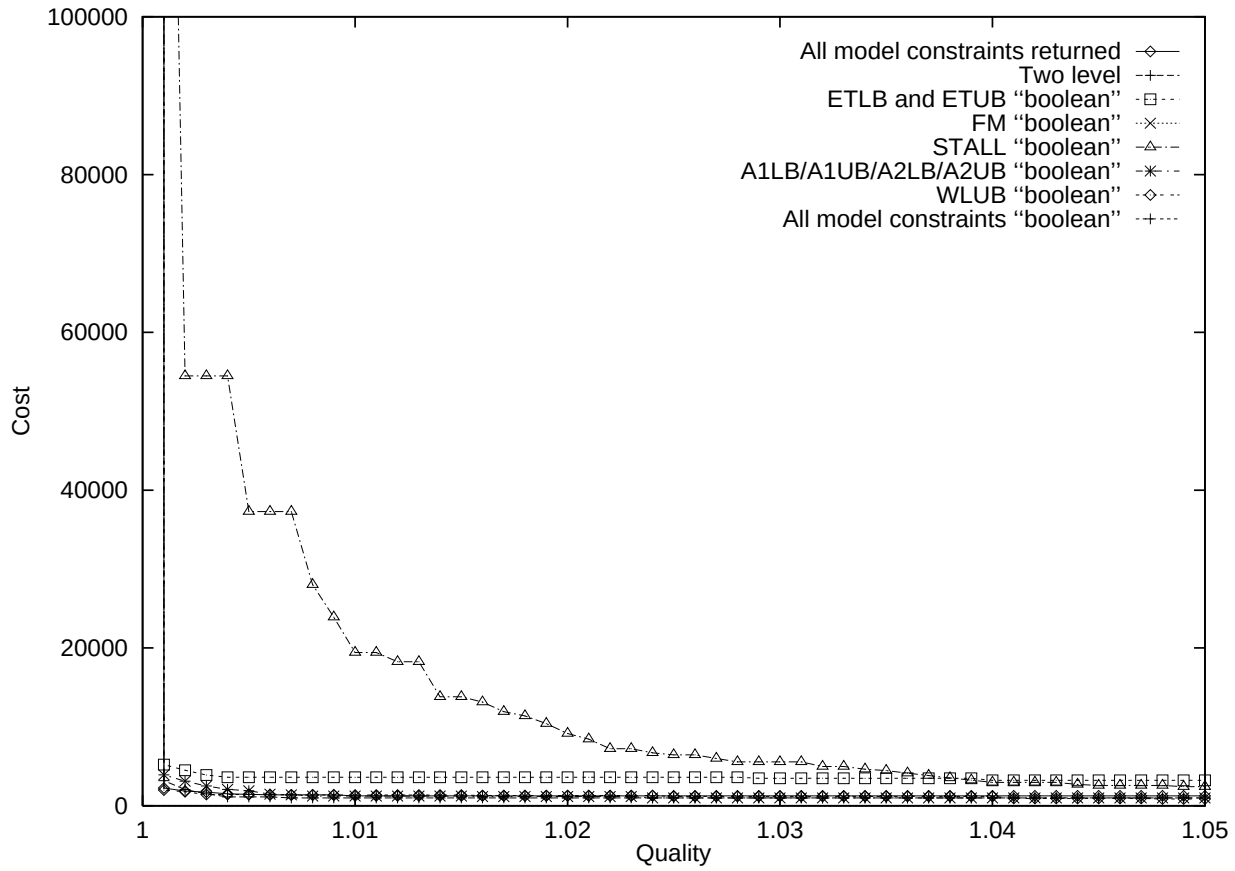

Figure 7: Cost to achieve a range of design qualities with 99% confidence. Quality is takeoff mass, normalized by the best takeoff mass found (Figure 5), so quality = 1.01 corresponds to Figure 6.

begins falling rapidly as stall occurs for higher angles of attack. The STALL constraint, which is active at our global optimum (see Figure 5) cuts off this function at its peak so that the lift function is monotonic where the constraint is satisfied. A more sophisticated simulator which modeled stall would not find better designs on the other side of the STALL constraint boundary — it would just find that the lift function ceased to be monotonic when the boundary was crossed. The ETLB constraint is also active at our global optimum (see Figure 5). In this case, the engine stops running when the throttle is too low. Modifying the engine model to correctly predict the sudden low temperatures and pressure produced by the engine when it stops running would not uncover better designs.

To test the effect of box size on our conclusions, we repeated our experiments in a larger box. Figure 4 shows the two "boxes" in the design space used in our experiments. The bigger box contains the smaller box, and the volume of the larger box is about 300 times greater than the volume of the smaller box. Figure 8 shows the performance of the various communication strategies in the larger box, and Figure 9 shows graphically the "Est. 99% Cost" to achieve a range of different design qualities. Search cost increases in the larger box, as expected, but model constraints still cost orders of magnitude less than the boolean strategy.

It is important to compare the performance of the "two-level" strategy combination for the two boxes. In the "small" box, the two-level approach was actually superior to the pure model constraints approach: it was slightly better to use a boolean strategy to find a feasible point before starting to use model constraints to find the optimum. The reverse was true in the big box, however: the pure model constraints approach was a factor of six less expensive then the two-level approach. These results are quite plausible, because the "start cost" data for "all boolean" combination indicates that the density of feasible points in the small box is about 1/300 while in the big box it is only 1/4800. The big box has such a small feasible region that the benefit of using model constraints to search for the feasible region outweighs the model constraints overhead, while in the smaller box random probes can find the feasible region cheaply enough that the overhead of using model constraint to find the feasible region is not justified. However, even in the small box model constraints are still extremely useful for searching within the feasible region in order to find an optimum.

To test the effect of the design goal on our conclusions, we repeated our experiments with a different goal. We used the same boxes as for the previous experiments, but instead the goal was to design the best aircraft for the mission shown in Figure 10. Figure 11 shows the best design found, which differs considerably from the optimal design for the previous mission. We performed the same set of experiments for this case, and the experimental data which appears in Figures 12, 13, 14, and 15 supports our previous conclusion that the model constraint communication strategy can cut search cost by an order of magnitude or more.

7 Related Work

[Gelsey 1995b] examines the types of modeling knowledge that are needed so that a simulator can be reliably invoked by another program and describes algorithms for detecting assumption violations and other problems that might lead to low-quality or unreliable simulation results, but strategies for communicating information about modeling fail-

Strategy Combination	Success	Start Cost	Opt. Cost	Est. 99% Cost
All model constraints returned	55/74	48	58376	2674
ETLB and ETUB "boolean"	14/74	6979	124796	39102
FM "boolean"	0/74	879	128618	$\gg$ 592317
STALL "boolean"	15/74	21669	78508	27520
A1LB/A1UB/A2LB/A2UB "boolean"	40/74	280	176113	14114
WLUB "boolean"	60/74	2181	58976	2285
All model constraints "boolean"	0/74	354761	80835	$\gg$ 1992408
Two level	54/74	301917	56198	17034

Figure 8: Performance of the various strategy combinations in a bigger box

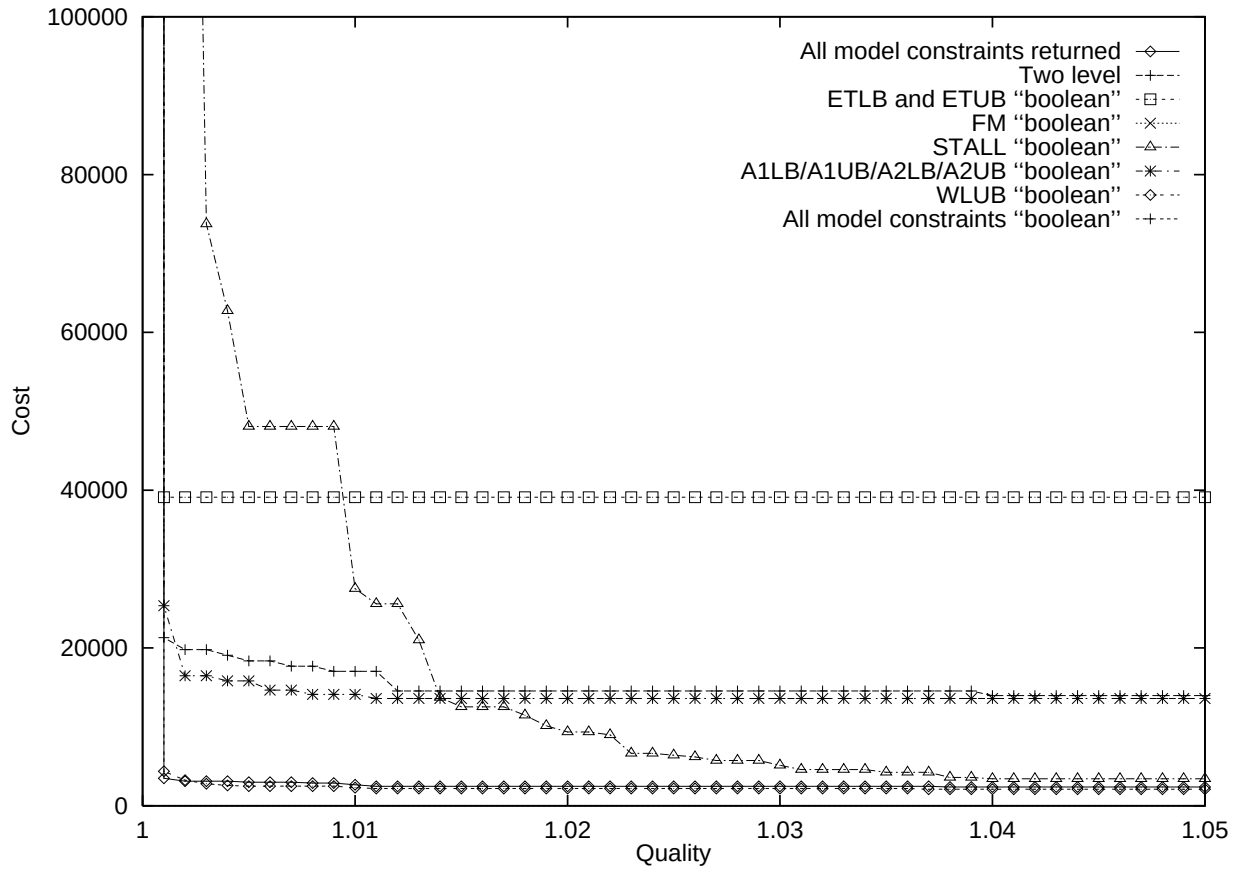

Figure 9: Cost to achieve a range of design qualities with 99% confidence. Quality is takeoff mass, normalized by the best takeoff mass found (Figure 5), so quality = 1.01 corresponds to Figure 8.

Phase	Mach	Altitude (ft.)	Duration (min.s)	comment
1	0.227	0	5	"takeoff"
2	0.85	40,000	50	subsonic cruise (over land)
3	2.0	60,000	225	supersonic cruise (over ocean)

capacity: 70 passengers.

Figure 10: Another mission specification

Design Parameters:	
engine size	1.146
wing area	3690 sq. ft.
wing aspect ratio	1.089
fuselage taper length	130.1 ft.
effective structural thickness over chord	2.728
wing sweep over design mach angle	1.235
wing taper ratio	0
fuel annulus width	0
Objective Function:	
Takeoff Mass	134.8 tonnes
Model Constraints:	
ETUB	-2.89
ETLB	-18.19
A1LB	-1.83
A1UB	-2.17
A2LB	-2.03
A2UB	-7.97
WLUB	-143.8
FM	-0.00038 tonnes
STALL	0
Design Constraint:	
PASS	-2

Figure 11: Best design found for the 2nd mission (Figure 10)

Strategy Combination	Success	Start Cost	Opt. Cost	Est. 99% Cost
All model constraints returned	62/74	13	36204	1238
ETLB and ETUB "boolean"	57/74	1275	65556	2827
FM "boolean"	1/74	31	65105	297930
STALL "boolean"	36/74	1681	50847	4904
A1LB/A1UB/A2LB/A2UB "boolean"	65/74	55	40377	1194
WLUB "boolean"	64/74	227	34899	1092
All model constraints "boolean"	0/74	6576	67046	$\gg$ 336745
Two level	64/74	4307	34477	1205

Figure 12: Performance of the various strategy combinations for the 2nd mission

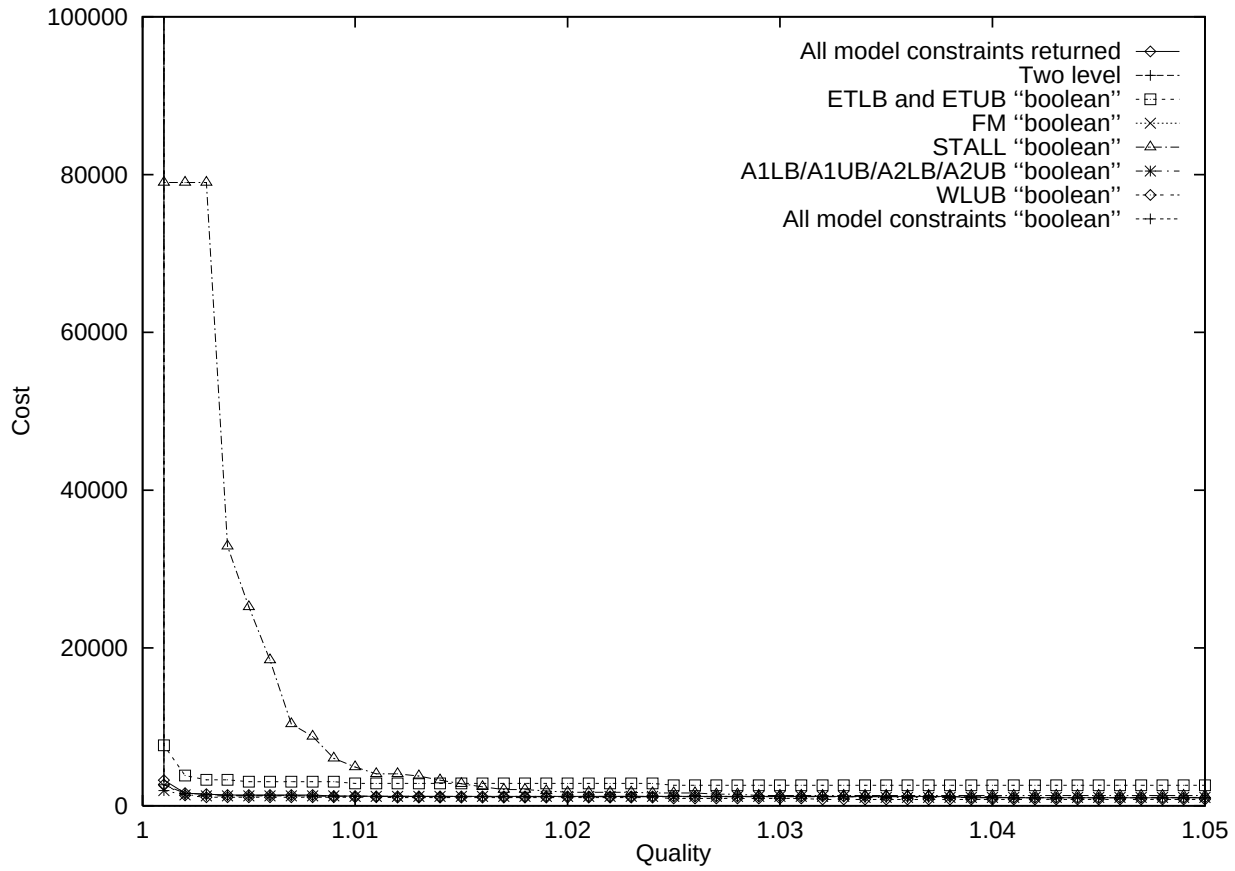

Figure 13: Cost to achieve a range of design qualities with 99% confidence. Quality is takeoff mass, normalized by the best takeoff mass found (Figure 11), so quality = 1.01 corresponds to Figure 12.

Strategy Combination	Success	Start Cost	Opt. Cost	Est. 99% Cost
All model constraints returned	48/74	41	57607	3429
ETLB and ETUB "boolean"	13/74	5616	145770	48765
FM "boolean"	0/74	162	93146	$\gg$ 426789
STALL "boolean"	39/74	7602	64000	5951
A1LB/A1UB/A2LB/A2UB "boolean"	34/74	342	131652	13352
WLUB "boolean"	51/74	1420	56171	3066
All model constraints "boolean"	0/74	133265	117224	$\gg$ 1145732
Two level	49/74	231396	48049	16025

Figure 14: Performance of the various strategy combinations for the 2nd mission in the bigger box

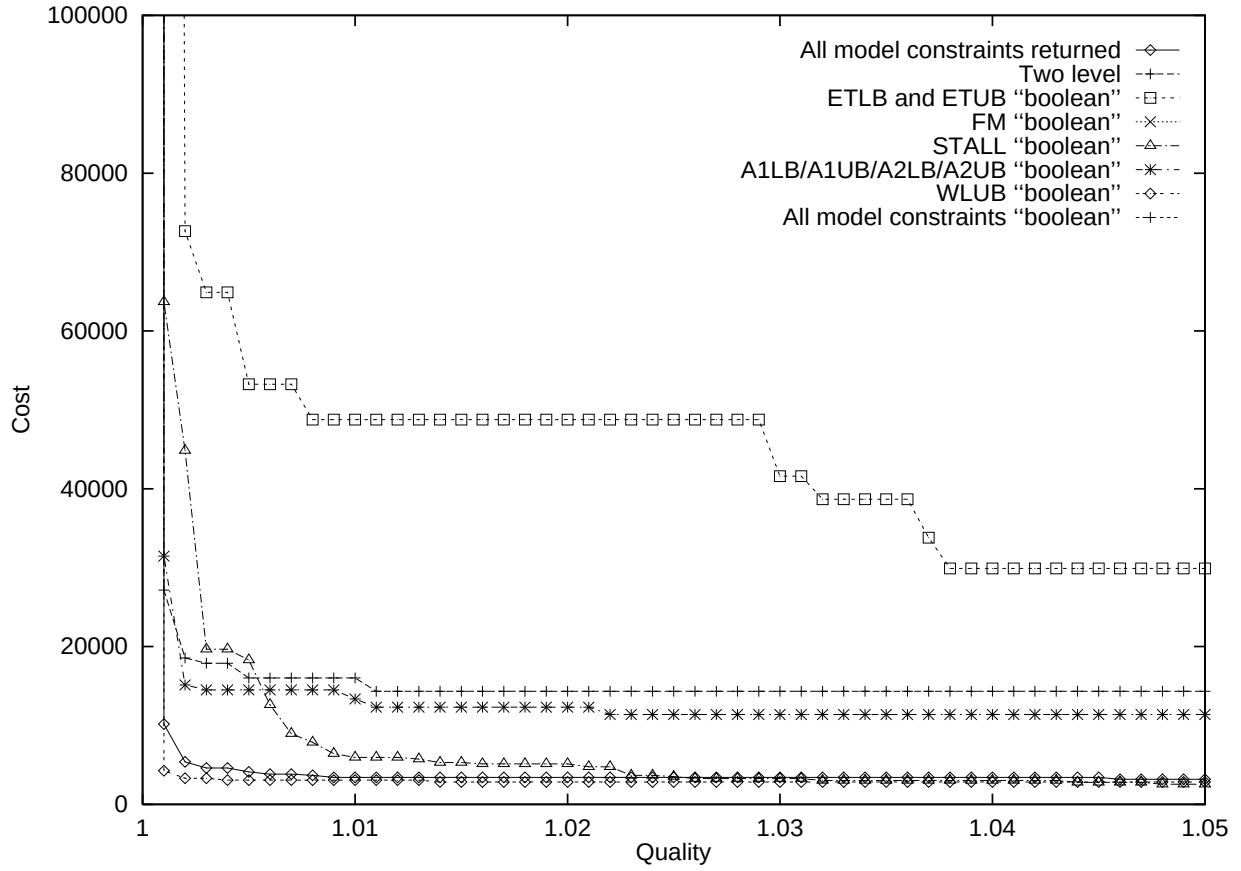

Figure 15: Cost to achieve a range of design qualities with 99% confidence. Quality is takeoff mass, normalized by the best takeoff mass found (Figure 11), so quality = 1.01 corresponds to Figure 14.

ures to an automated design systems are not discussed. [Forbus and Falkenhainer 1990, Forbus and Falkenhainer 1992, Forbus and Falkenhainer 1995] discuss the use of qualitative simulation to check the quality of numerical simulation results, but here strategies for communicating information about modeling failures to an automated design systems are also not discussed.

Other automated intelligent controllers for numerical simulators are described in [Gelsey 1991, Gelsey 1995a, Sacks 1991, Yip 1991, Zhao 1994], but these do not address the issue of model and simulation quality assurance.

Intelligent monitoring for complex systems has received considerable attention (e.g., [Dvorak and Kuipers 1991]), but this work has focused on diagnosis of problems in dynamically changing physical systems as opposed to problems in the execution of computational algorithms which are attempting to simulate the behavior of physical systems.

A great deal of work has been done in the area of numerical optimization algorithms [Gill *et al.* 1981, Vanderplaats 1984, Peressini *et al.* 1988, Moré and Wright 1993, Papalambros and Wilde 1988], though not much has been published about the particular difficulties of attempting to optimize functions defined by large "real-world" numerical simulators. A number of research efforts have combined AI techniques with numerical optimization [Ellman *et al.* 1993, Schwabacher *et al.* 1994, Schwabacher *et al.* 1996, Tong *et al.* 1992, Powell 1990, Bouchard *et al.* 1988, Bouchard 1992, Sobieszczanski-Sobieski *et al.* 1985, Agogino and Almgren 1987, Williams and Cagan 1994, Hoeltzel and Chieng 1987, Cerbone 1992], but have not addressed the issue of model and simulation quality assurance.

8 Limitations and Future Work

A limitation of the model constraints communication strategy is the need to implement fairly well-behaved model constraint functions for all model assumptions. Implementing the model constraint functions was not too difficult for our conceptual design of aircraft domain, but investigating the difficulty of implementing model constraints in other domains is an important area for future work.

Our experiments have been performed in a domain in which the global optimum has a fairly large "basin of attraction", so that a local optimization method like Sequential Quadratic Programming will give a high confidence of finding the global optimum if started from a small number of random starting points. For domains in which this property fails to hold, global optimization methods such as Simulated Annealing will often be preferable. Such methods would not typically be able to make direct use of model constraint functions, so for such a domain investigating the "model penalties" communication strategy described in Section 2 might be worthwhile area for future work.

9 Conclusion

Automated search of a space of candidate designs seems an attractive way to improve the traditional engineering design process. To make this approach work, however, the automated design system must include both knowledge of the modeling limitations of the method used

to evaluate candidate designs and also an effective way to use this knowledge to influence the search process. We suggest that a productive approach is to include this knowledge by implementing a set of *model constraint* functions which measure how much each modeling assumptions is violated, and to influence the search by using the values of these model constraint functions as constraint inputs to a standard constrained nonlinear optimization numerical method. Our experiments indicate that our model constraint communication strategy can decrease the cost of design space search by one or more orders of magnitude.

10 Acknowledgments

We thank our aircraft design expert, Gene Bouchard of Lockheed, for his invaluable assistance in this research. We thank our programmer, Keith Miyake, for his effort in implementing large parts of the software described in this paper. This research was partially supported by NASA under grant NAG2-817 and is also part of the Rutgers-based HPCD (Hypercomputing and Design) project supported by the Advanced Research Projects Agency of the Department of Defense through contract ARPA-DABT 63-93-C-0064.

References

[Agogino and Almgren 1987] A. M. Agogino and A. S. Almgren. Techniques for integrating qualitative reasoning and symbolic computing. *Engineering Optimization*, 12:117–135, 1987.

[Bouchard *et al.* 1988] E. E. Bouchard, G. H. Kidwell, and J. E. Rogan. The application of artificial intelligence technology to aeronautical system design. In *AIAA/AHS/ASEE Aircraft Design Systems and Operations Meeting*, Atlanta, Georgia, September 1988. AIAA-88-4426.

[Bouchard 1992] E. E. Bouchard. Concepts for a future aircraft design environment. In *Proceedings, 1992 Aerospace Design Conference*, Irvine, CA, February 1992. AIAA-92-1188.

[Cerbone 1992] G. Cerbone. Machine learning in engineering: Techniques to speed up numerical optimization. Technical Report 92-30-09, Oregon State University Department of Computer Science, 1992. Ph.D. Thesis.

[Dvorak and Kuipers 1991] Daniel Dvorak and Benjamin Kuipers. Process monitoring and diagnosis. *IEEE Expert*, 6(3):67–74, June 1991.

[Ellman *et al.* 1993] T. Ellman, J. Keane, and M. Schwabacher. Intelligent model selection for hillclimbing search in computer-aided design. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, Washington, D.C., 1993.

[Forbus and Falkenhainer 1990] Kenneth D. Forbus and Brian Falkenhainer. Self-explanatory simulations: An integration of qualitative and quantitative knowledge. In

Proceedings, Eighth National Conference on Artificial Intelligence, pages 380–387, Boston, MA, 1990.

[Forbus and Falkenhainer 1992] Kenneth D. Forbus and Brian Falkenhainer. Self-explanatory simulations: Scaling up to large models. In *Proceedings, Tenth National Conference on Artificial Intelligence*, San Jose, CA, 1992.

[Forbus and Falkenhainer 1995] Kenneth D. Forbus and Brian Falkenhainer. Scaling up self-explanatory simulations: Polynomial-time compilation. In *Proceedings, Fourteenth Internationnal Joint Conference on Artificial Intelligence*, Montreal, Quebec, Canada, 1995.

[Gelsey and Smith 1995] Andrew Gelsey and Don Smith. A search space toolkit. In *Proceedings, 11th IEEE Conference on Artificial Intelligence Applications*, pages 117–123, Los Angeles, CA, February 1995.

[Gelsey *et al.* 1996] Andrew Gelsey, Don Smith, Mark Schwabacher, Khaled Rasheed, and Keith Miyake. A search space toolkit. *Decision Support Systems, special issue on Unification of Artificial Intelligence with Optimization*, 1996. To appear.

[Gelsey 1991] Andrew Gelsey. Using intelligently controlled simulation to predict a machine's long-term behavior. In *Proceedings, Ninth National Conference on Artificial Intelligence*, pages 880–887, Cambridge, MA, July 1991.

[Gelsey 1995a] Andrew Gelsey. Automated reasoning about machines. *Artificial Intelligence*, 74(1):1–53, March 1995.

[Gelsey 1995b] Andrew Gelsey. Intelligent automated quality control for computational simulation. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM)*, 9(5):387–400, November 1995.

[Gill *et al.* 1981] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimization.* Academic Press, London ; New York, 1981.

[Hoeltzel and Chieng 1987] D. Hoeltzel and W. Chieng. Statistical machine learning for the cognitive selection of nonlinear programming algorithms in engineering design optimization. In *Advances in Design Automation*, Boston, MA, 1987.

[Lawrence *et al.* 1995] C. Lawrence, J. Zhou, and A. Tits. User's guide for CFSQP version 2.3: A C code for solving (large scale) constrained nonlinear (minimax) optimization problems, generating iterates satisfying all inequality constraints. Technical Report TR-94-16r1, Institute for Systems Research, University of Maryland, August 1995.

[Moré and Wright 1993] Jorge J. Moré and Stephen J. Wright. *Optimization Software Guide.* SIAM, Philadelphia, 1993.

[Papalambros and Wilde 1988] P. Papalambros and J. Wilde. *Principles of Optimal Design.* Cambridge University Press, New York, NY, 1988.

[Peressini *et al.* 1988] Anthony L. Peressini, Francis E. Sullivan, and J. J. Uhl, Jr. *The Mathematics of Nonlinear Programming.* Springer-Verlag, New York, 1988.

[Powell 1990] D. Powell. Inter-GEN: A hybrid approach to engineering design optimization. Technical report, Rensselaer Polytechnic Institute Department of Computer Science, December 1990. Ph.D. Thesis.

[Sacks 1991] Elisha P. Sacks. Automatic analysis of one-parameter ordinary differential equations by intelligent numeric simulation. *Artificial Intelligence*, 48(1):27–56, February 1991.

[Schwabacher and Gelsey 1996] M. Schwabacher and A. Gelsey. Intelligent gradient-based search of incompletely defined design spaces. Technical Report HPCD-TR-38, Department of Computer Science, Rutgers University, New Brunswick, NJ, 1996. ftp://ftp.cs.rutgers.edu/pub/technical-reports/hpcd-tr-38.ps.Z.

[Schwabacher *et al.* 1994] M. Schwabacher, H. Hirsh, and T. Ellman. Learning prototype-selection rules for case-based iterative design. In *Proceedings of the Tenth IEEE Conference on Artificial Intelligence for Applications*, San Antonio, Texas, 1994.

[Schwabacher *et al.* 1996] M. Schwabacher, T. Ellman, H. Hirsh, and G. Richter. Learning to choose a reformulation for numerical optimization of engineering designs. In *Artificial Intelligence in Design Conference*, June 1996. To appear.

[Sobieszczanski-Sobieski *et al.* 1985] J. Sobieszczanski-Sobieski, B. B. James, and A. R. Dovi. Structural optimization by multilevel decomposition. *AIAA Journal*, 23(11):1775–1782, November 1985.

[Tong *et al.* 1992] Siu Shing Tong, David Powell, and Sanjay Goel. Integration of artificial intelligence and numerical optimization techniques for the design of complex aerospace systems. In *Proceedings, 1992 Aerospace Design Conference*, Irvine, CA, February 1992. AIAA.

[Vanderplaats 1984] Garret N. Vanderplaats. *Numerical Optimization Techniques for Engineering Design : With Applications.* McGraw-Hill, New York, 1984.

[Williams and Cagan 1994] Brian C. Williams and Jonathan Cagan. Activity analysis: the qualitative analysis of stationary points for optimal reasoning. In *Proceedings, 12th National Conference on Artificial Intelligence*, pages 1224–1230, Seattle, Washington, August 1994.

[Yip 1991] Kenneth Yip. Understanding complex dynamics by visual and symbolic reasoning. *Artificial Intelligence*, 51(1–3):179–221, October 1991.

[Zhao 1994] Feng Zhao. Extracting and representing qualitative behaviors of complex systems in phase space. *Artificial Intelligence*, 69(1-2):51–92, 1994.