



Using modeling knowledge to guide design space search

Andrew Gelsey^{*}, Mark Schwabacher¹, Don Smith²

Computer Science Department, Rutgers University, New Brunswick, NJ 08903, USA

Received 26 March 1997; received in revised form 5 January 1998

Abstract

Automated search of a space of candidate designs is an attractive way to improve the traditional engineering design process. To make this approach work, however, an automated design system must include both knowledge of the modeling limitations of the method used to evaluate candidate designs and an effective way to use this knowledge to influence the search process. We argue that a productive approach is to include this knowledge by implementing a set of *model constraint* functions which measure how much each modeling assumption is violated. The search is then guided by using the values of these model constraint functions as constraint inputs to a standard constrained nonlinear optimization numerical method. A key result of our work is a successful demonstration of the application of AI techniques to an important engineering problem. In an empirical study of parametric conceptual aircraft design, we observed a cost improvement of two orders of magnitude. The principal contribution of our work is a new design optimization methodology which makes explicit the interaction between models of artifacts, and validity models of artifact models. © 1998 Elsevier Science B.V. All rights reserved.

Keywords: Design; Engineering; Model; Optimization; Physics; Search; Aircraft; Constraint; Numerical

1. Introduction

Automated search is not commonly used in real-world design tasks, in spite of the widely held belief that search automation should improve the traditional engineering process by producing better designs in less time. We explain why automated design

^{*} Corresponding author. Email: gelsey@cs.rutgers.edu.

¹ Email: schwabac@cs.rutgers.edu.

² Email: dsmith@cs.rutgers.edu.

is not used (simulators do not know their limits), and show how this problem can be overcome using *model constraints* that model the limitations of the simulators. The principal contribution of our work is a new design optimization methodology which makes explicit the interaction between models of artifacts, and validity models of artifact models. We show an application of AI techniques to an important engineering problem, presenting an effective compromise between, on the one hand, using the results of a simulator as given, or, on the other hand, completely re-engineering it declaratively. Re-engineering has been successful in previous research (e.g. [9], see Section 7) but may be prohibitively expensive.

A key contribution of our work is a successful demonstration of the application of AI techniques to an important engineering problem. In an empirical study of parametric conceptual aircraft design, we observed a cost improvement of two orders of magnitude. The details of this study appear later in this paper, as well as some evidence that such improvements may transfer to other domains.

If the design process is automated, each step of the 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 will use their domain knowledge 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 will not 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 does not want to design a plane that stalls during normal operation, so he does not need a detailed prediction of stall behavior. In our interactions with aircraft industry expert design engineers, we have found that human experts may in fact be quite satisfied with an incomplete lift model. Their domain knowledge gives them the ability to recognize "impossibly high" lift coefficients and thus realize that the candidate design would actually stall and should be discarded.

However, the simulator may be invoked by another program such as an automated search procedure rather than by a human expert. In this case it is quite likely that in exploring the design space, the automated search procedure will examine designs which violate the simulator's assumptions. 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. This problem is one reason that automated search is not commonly used in "real-world" engineering design.

In our earlier work [16], we have investigated the types of modeling knowledge that are needed so that a simulator can be reliably invoked by another program. We have also described algorithms for detecting assumption violations and other problems that

might lead to incorrect or unreliable simulation results. In the present article, we address the question of how information about model assumption violations can be effectively communicated to an automated search procedure. This communication allows the search procedure to focus its search for good designs within the subset of the design space which contains candidate designs that do not violate model assumptions.

2. Communication strategies

As mentioned above, the focus of this research is the effective communication of information about model assumption violation between a simulator and an automated search procedure which is exploring a space of candidate designs. In our experimental work we have used a gradient-based constrained optimization algorithm as our search procedure. (This search procedure is further described in Section 4.) However, before discussing the details of this particular search method, we would like to present the following more general list of strategies. These strategies can be used for communication of information about model assumption violation between a simulator and an automated search procedure which is exploring a space of candidate designs:

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 article 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 article 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 do not investigate the model penalties strategy in this article, but discuss possible uses for it in Section 8.

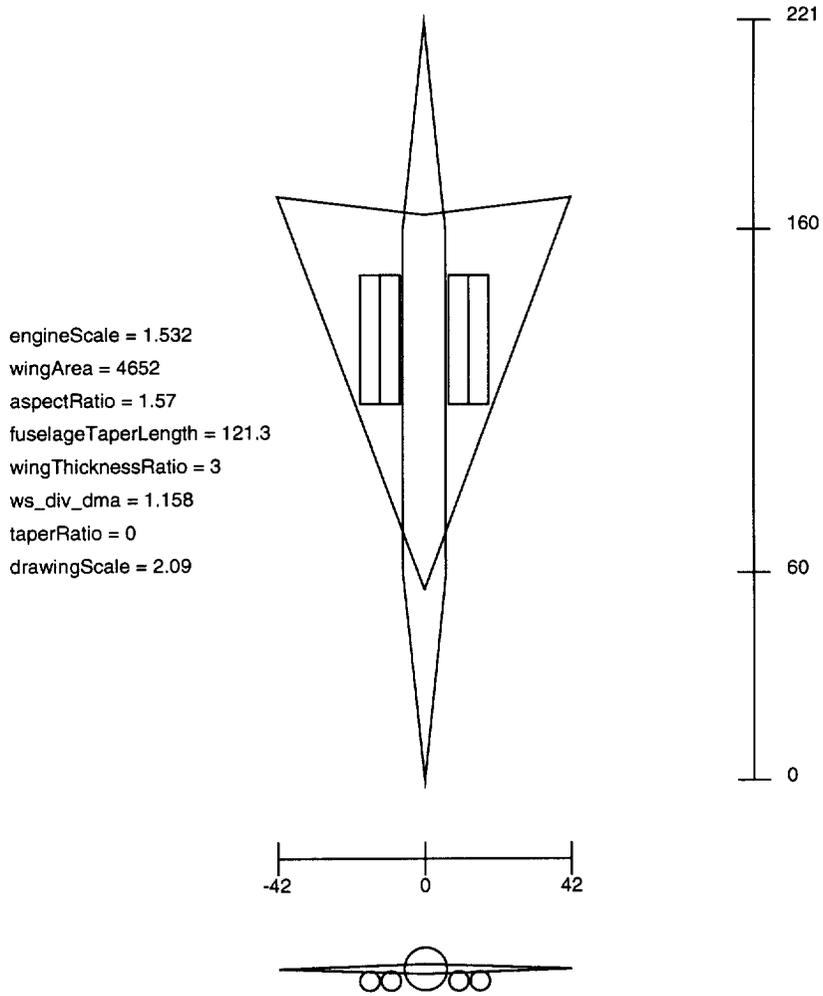


Fig. 1. Supersonic transport aircraft designed by our system (dimensions in feet).

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

Fig. 2. Mission specification for aircraft in Fig. 1.

3. Aircraft design

We have pursued our investigation in the domain of conceptual design of supersonic transport aircraft. However, in our design task the key design variables have already been identified (in collaboration with an aircraft industry design expert) so a more precise characterization of our problem might be “parametric design at a system level of abstraction”.

Fig. 1 shows a diagram of an airplane designed to fly the mission shown in Fig. 2. This mission is for a supersonic passenger transport, so a key requirement is the passenger capacity (70 persons in this case). The mission has three key phases: a short, low-speed, ground level phase to test takeoff capability, a subsonic cruise phase representing travel over land where supersonic flight is prohibited, and finally a supersonic cruise phase corresponding to an ocean crossing.

For conceptual design, we represent an aircraft just by a set of values for major aircraft design variables such as wing area, aspect ratio, engine size, etc. The experiments in this article use eight such design variables, which are listed later in Section 6. In our current implementation, the design 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. Takeoff mass of a particular aircraft design for a particular mission is computed as follows:

1. Compute “dry” mass using historical data to estimate the weight of the aircraft as a function of the design variables 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.

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

4. Design Associate

Fig. 3 shows a block diagram of our automated conceptual design system. The design system has two major components: the Design Associate (DA), which searches the

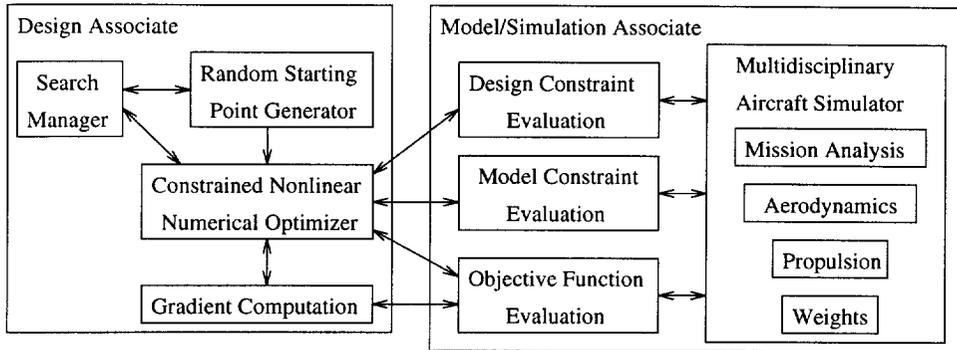


Fig. 3. Automated design system block diagram.

space of candidate designs, and the Model/Simulation Associate (MSA), which the DA uses to evaluate the quality of candidate designs. Unlike the discrete search spaces more commonly studied by AI researchers, the search space for the aircraft conceptual design problem involves design variables such as wing area or aspect ratio which can be varied continuously throughout an interval of possible values. To search this space, the DA uses a constrained nonlinear numerical optimizer, which varies the set of continuous design variables to minimize³ a nonlinear objective function subject to a set of nonlinear equality and inequality constraints. As mentioned previously, the nonlinear objective function to be minimized is the takeoff mass required for a particular candidate aircraft design to fly a particular mission. The constraints are computed within the Model/Simulation Associate and are described in Section 5.

The data flow in Fig. 3 is as follows:

- The search manager (in conjunction with the random starting point generator) passes to the constrained nonlinear optimizer a design represented as a vector of real numbers, the values of the design variables. The optimizer uses this initial design as a starting point and later passes back an improved design using the same representation.
- The constrained nonlinear optimizer (directly or via the gradient computation module) passes to the evaluation modules a design, again represented as a vector of values of the design variables. The design constraint evaluation module passes back a vector of real numbers representing the values of the design constraints. The model constraint module does the same for model constraints, and the objective function evaluation module passes back a scalar value for design quality, which however is only meaningful if all the constraints are satisfied.
- The evaluation modules pass on to the simulator the design passed to them, and the simulator passes back a complete set of simulation results from which each evaluation module then extracts the data it needs.

³To instead *maximize* the objective function, just multiply it by -1 and minimize.

The numerical optimizer used in this article is CFSQP⁴ [23], a state-of-the-art implementation of the Sequential Quadratic Programming method.⁵ Sequential Quadratic Programming is a quasi-Newton method that solves a nonlinear constrained optimization problem by fitting a sequence of quadratic programs⁶ to it, and then solving each of these problems using a quadratic programming method [20,43].

CFSQP begins its search of the design space at a particular point (i.e., a particular candidate design), and typically a set of searches from several starting points will not all terminate at or near a single global optimum. Therefore, the DA includes a random starting point generator so that CFSQP can be started from several different candidate designs, improving the likelihood of finding an excellent design.

In order to handle unevaluable points (i.e., points each of whose objective function was assigned the boolean strategy standard “very bad” value), the DA includes methods for “intelligent” gradient computation. The gradients used by CFSQP are computed by using a set of rules that specify how to compute gradients with reasonable accuracy in the presence of unevaluable points. For example, if the DA evaluates three candidate designs in order to compute a component of the gradient using a central difference formula, and if one of the points is unevaluable, then the DA ignores the unevaluable point and uses the other two points in a forward difference formula. The DA’s rules for gradient computation are described in detail in [35]. 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 unevaluable points exist even when using model constraints. In the experiments reported in this article, with the boolean strategy, 76% of the points encountered were unevaluable, and with the model constraints strategy, 4% of the points encountered were unevaluable.⁷

5. Model/Simulation Associate

Our automated design system (Fig. 3) has two major components: the Design Associate (DA), described above, and the Model/Simulation Associate (MSA), which the DA uses to evaluate the quality of candidate designs. The MSA includes a multidisciplinary⁸ simulator which it uses to evaluate candidate designs, and the MSA is responsible for communicating to the DA information about the quality of a candidate

⁴ CFSQP stands for “C code for Feasible Sequential Quadratic Programming”.

⁵ Earlier we tried doing optimization in this domain using several different optimization packages, and found that we obtained the best results when using CFSQP.

⁶ A quadratic program consists of a quadratic objective function to be optimized, and a set of linear constraints.

⁷ 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 article.

⁸ We call the simulator *multidisciplinary* because it contains code to evaluate the aircraft using several engineering disciplines, including weights, aerodynamics, and propulsion.

design, how well other design constraints are satisfied, and how well the simulator's modeling assumptions are satisfied. Our current version of the MSA implements two of the communication strategies described in Section 2: the boolean strategy and the model constraints strategy. When the MSA runs the multidisciplinary aircraft simulator to evaluate a candidate design, it computes values for both the objective function and a set of design constraint functions and model constraint functions. To implement the model constraints communication strategy, the MSA just returns values for all these functions to the DA. To implement the boolean communication strategy, the MSA returns only a value for the objective function, which is computed as follows:

- if one or more constraints are violated, the MSA sends a standard “very bad” value to the DA as the value of the objective function;
- otherwise, the MSA sends to the DA the value of the objective function computed by the simulator, which will be the takeoff mass the candidate design requires to complete the mission.

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

MaxThrottle = (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 MaxThrottle will indicate the extent to which the engine model assumptions are violated.

MinThrottle = (minimum throttle setting allowed for engine) – (minimum throttle required during mission simulation).

DragTableL1, DragTableU1, DragTableL2, DragTableU2: Similar to above—violation of bounds for a two-dimensional table of experimental data on supersonic drag.

LOAD = (maximum wing loading during mission simulation) – (maximum wing loading simulator can validly model).

FUEL = (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 will not 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 variables as their values change sign. This smoothness 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.

We believe that if, for a given model constraint, a design space can be partitioned into a small number of regions in which the constraint holds or does not hold, then it will usually be possible to formulate a model constraint which will smoothly approach zero as the boundaries of the “good” regions are approached. If, however, throughout the design space a model constraint oscillates with high frequency between satisfaction and violation, then it probably will not be smooth. In this case the model will almost certainly need to be revised in order to be useful for design.

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

$$\text{PASS} = (\text{passenger capacity required for the mission}) - (\text{passenger capacity available with current design variables}).$$

To clarify the distinction between model constraints and design constraints, we note the following differences:

- Design constraints can be extracted directly from design goals, while formulating model constraints requires carefully examining the underlying assumptions of the model on which the simulator is based.
- Design constraints can be violated without reducing the 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. This design may appear to be a very efficient aircraft, much better than the best physically plausible design, but 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. If the model constraint violations are ignored by the null strategy, the region where the model constraint is violated may include apparent local optima of the objective function. Even worse, there may be 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 it 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 variables 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),
- (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),

This set of design variables was chosen in collaboration with an aircraft industry design expert.⁹ However, in these experiments we omitted discrete parameters such as number of engines which did not fit well with our continuous nonlinear programming search method. Since there are only a small number of choices, in practice our continuous design methodology could simply be repeated several times using different numbers of engines, and the best of these four or five designs could be chosen. A more general approach would be the use of mixed integer/continuous programming techniques as a search procedure, but that would require significant additional research.

Fig. 4 shows the two subsets we explored in the design space defined by these design variables. Each is an eight-dimensional “box” which is the product of the intervals of values allowed for the eight design variables. The volume of the “big box” is about 300 times larger than the volume of the “small box”.¹⁰

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 MinThrottle and MaxThrottle to the optimizer, but for candidate designs where the engine table constraints were violated (MinThrottle or MaxThrottle positive), use the “boolean” strategy and return a standard “very large” value for takeoff mass.
- (3) Return values of all model constraint functions except DragTableL1, DragTableU1, DragTableL2, and DragTableU2 to the optimizer; use “boolean” strategy for points which required extrapolation outside the aerodynamics table bounds.

⁹ Dr. Gene Bouchard of Lockheed.

¹⁰ This nondimensional volume ratio is the product of the ratios of the ranges of the design variables,

$$300 \approx \prod_{i \in \text{design variables}} \frac{b_i}{s_i},$$

where b_i is the range of design variable i in the larger box and s_i is the range of design variable i in the smaller box. Since for a given i , b_i and s_i have the same units, their ratio is nondimensional and thus the product of all these ratios is itself also nondimensional. This nondimensional volume ratio will be preserved by linear transformations of the design variables, for example by changes in units of measurement, though the volume ratio could be changed by a problem reformulation which transformed the set of design variables in a nonlinear fashion.

Design variable	Small box		Big box	
	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

Fig. 4. Subsets of design space explored.

- (4) Return values of all model constraint functions except FUEL, 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 LOAD, which is “boolean”.
- (7) Use the “boolean” communication strategy for all model constraint functions.
- (8) A two-level approach in which the “boolean” communication strategy is used to find a feasible point (i.e., a design for which all constraints are satisfied), and then all model constraints are used to find an optimum.

We chose these combinations so we could test the impact of each type of model constraint by comparing results with all model constraints to results with a given type left out.

For each strategy combination, our system randomly chose 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).¹¹ 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 Fig. 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 Fig. 5, and a diagram of this aircraft appears in Fig. 1.

The performance of the strategy combinations is shown in a table in Fig. 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 (Fig. 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.¹² The “Est.

¹¹ 74 is not a “magic” number; it was just a convenient choice given available disk space.

¹² As mentioned earlier, a complete mission simulation requires about 1/4 second of CPU time on a DEC Alpha 250 4/266 workstation.

<i>Design variables:</i>	
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
<i>Objective function:</i>	
Takeoff Mass	167.4 tonnes
<i>Model constraints:</i>	
MaxThrottle	-41.57
MinThrottle	-0.76
DragTableL1	-2.2
DragTableU1	-1.8
DragTableL2	-1.5
DragTableU2	-8.5
LOAD	-149.8
FUEL	-0.0011 tonnes
STALL	0
<i>Design constraint:</i>	
PASS	-2

Fig. 5. Best design found for mission of Fig. 2.

Strategy combination	Success	Start cost	Opt. cost	Est. 99% cost
All model constraints returned	65/74	16	42375	1252
Min/MaxThrottle "boolean"	52/74	3203	67158	3609
FUEL "boolean"	0/74	603	99215	≫ 456565
STALL "boolean"	18/74	5441	81566	19427
DragTable* "boolean"	67/74	57	47042	1242
LOAD "boolean"	62/74	721	39404	1372
All model constraints "boolean"	0/74	21946	75804	≫ 447106
Two level	72/74	18098	39389	990

Fig. 6. Performance of the various strategy combinations.

99% cost" column in Fig. 6 gives the estimated cost with each strategy combination to have a 99% chance of finding the global optimum. This cost 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

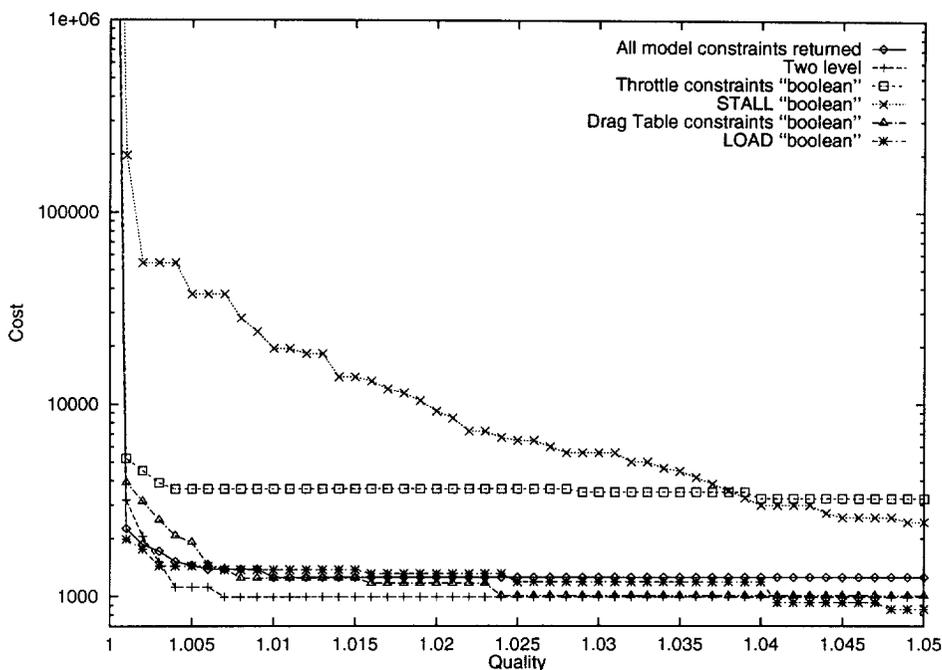


Fig. 7. Cost to achieve a range of design qualities with 99% confidence. Quality is takeoff mass, normalized by the best takeoff mass found (Fig. 5), so quality = 1.01 corresponds to Fig. 6. The 'FUEL "boolean"' and 'All model constraints "boolean"' strategy combinations do not appear as they did not find designs within 5% of the best takeoff mass found.

we estimate with the value in the "Success" column).¹³ As the table indicates, for the cases whose success was 0/74, the only information we can compute about the "Est. 99% cost" is that it will be greater than the cost we would have computed if the success rate had been 1/74. Fig. 7 shows graphically the "Est. 99% cost" to achieve a range of different design qualities.

¹³ Derivation of formula: $(1 - P_{\text{success}})$ is the probability that a single optimization will *not* find the global optimum, so $(1 - P_{\text{success}})^n$ is the probability that *none* of n optimizations will find the global optimum, and thus $(1 - (1 - P_{\text{success}})^n)$ is the probability that at least one of n optimizations *will* find the global optimum. To find the cost of P_{desired} , a given desired probability of finding the global optimum, solve

$$P_{\text{desired}} = 1 - (1 - P_{\text{success}})^n$$

for n , which gives

$$n = \log(1 - P_{\text{desired}}) / \log(1 - P_{\text{success}})$$

and finally multiply n by the average cost per optimization. We note that the computed value of n is not necessarily an integer, so a more precise calculation would round n up to the nearest integer.

The data in Fig. 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 Fig. 5). However, 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 to consider is 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 did not need as many constraints? Not necessarily. For example, lift initially rises as a function of angle of attack and later begins falling rapidly as stall occurs for higher angles of attack. The STALL constraint, which is active at our global optimum (see Fig. 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 MinThrottle constraint is also active at our global optimum (see Fig. 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. Fig. 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, as explained in an earlier footnote. Fig. 8 shows the performance of the various communication strategies in the larger box, and Fig. 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.

Strategy combination	Success	Start cost	Opt. cost	Est. 99% cost
All model constraints returned	55/74	48	58376	2674
Min/MaxThrottle "boolean"	14/74	6979	124796	39102
FUEL "boolean"	0/74	879	128618	≥ 592317
STALL "boolean"	15/74	21669	78508	27520
DragTable* "boolean"	40/74	280	176113	14114
LOAD "boolean"	60/74	2181	58976	2285
All model constraints "boolean"	0/74	354761	80835	≥ 1992408
Two level	54/74	301917	56198	17034

Fig. 8. Performance of the various strategy combinations in a bigger box.

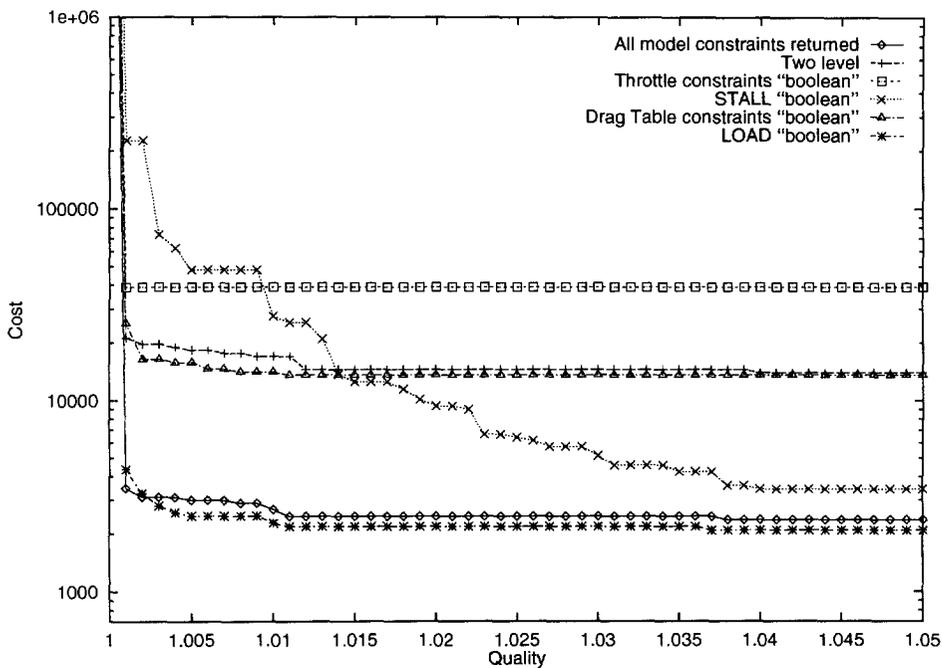


Fig. 9. Cost to achieve a range of design qualities with 99% confidence in a bigger box. Quality is takeoff mass, normalized by the best takeoff mass found (Fig. 5), so quality = 1.01 corresponds to Fig. 8.

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 than the two-level approach. These results are quite plausible, because the “start cost” data for “all boolean” combination indicates that the

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

Fig. 10. Another mission specification.

<i>Design variables:</i>	
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
<i>Objective function:</i>	
Takeoff Mass	134.8 tonnes
<i>Model constraints:</i>	
MaxThrottle	-2.89
MinThrottle	-18.19
DragTableL1	-1.83
DragTableU1	-2.17
DragTableL2	-2.03
DragTableU2	-7.97
LOAD	-143.8
FUEL	-0.00038 tonnes
STALL	0
<i>Design constraint:</i>	
PASS	-2

Fig. 11. Best design found for the second mission (Fig. 10).

density of feasible points in the small box is $74/21946$ ($\approx 1/300$) while in the big box it is only $74/354761$ ($\approx 1/4800$). 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 constraints 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.

Note that in our smaller box the feasible region forms a much higher proportion of the volume than is the case in our larger box. This observation suggests that by using a much smaller box, the feasible region will fill so much of box that specialized

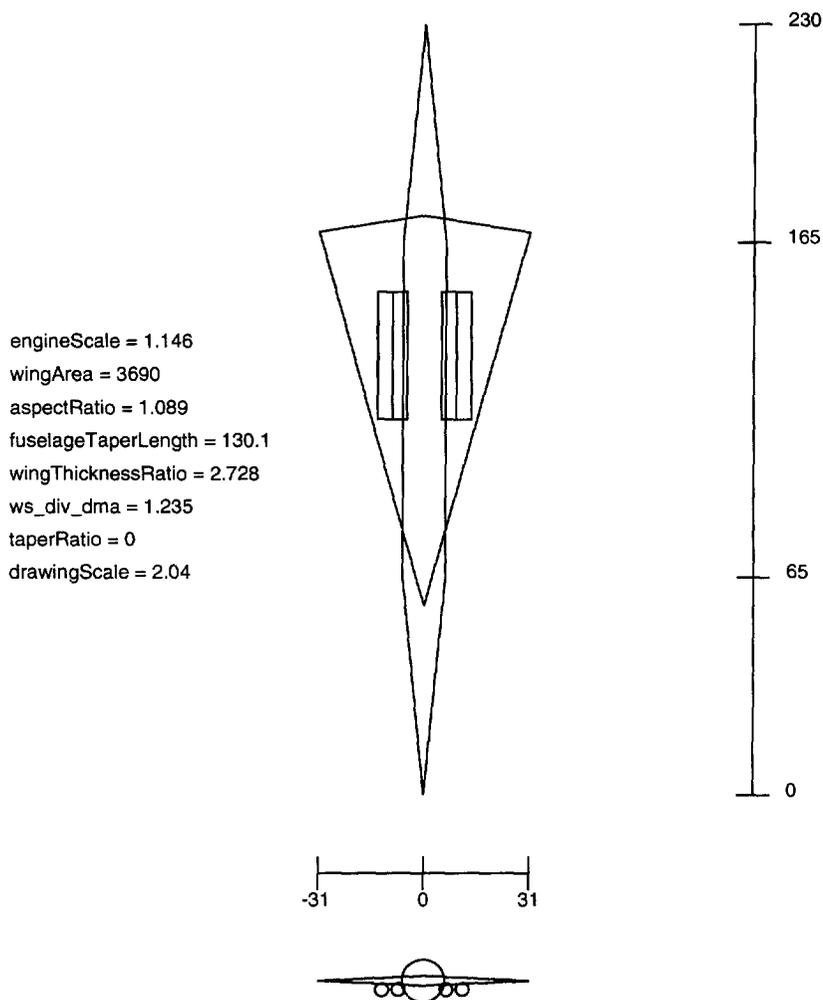


Fig. 12. Aircraft designed by our system for the second mission (dimensions in feet).

treatment of modeling issues becomes unnecessary. We suspect that in practice today optimization is often done by using human expertise and historical experience to greatly restrict design variable ranges so that optimization will be done in such a small box that model constraints are not needed. However, this approach is brittle in the sense that it will not scale to allow the sort of wide-ranging exploration of the design space which is possible with our explicit approach to modeling issues.

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 Fig. 10. Fig. 11 shows the best design found, which differs considerably from the optimal design for the

Strategy combination	Success	Start cost	Opt. cost	Est. 99% cost
All model constraints returned	62/74	13	36204	1238
Min/MaxThrottle "boolean"	57/74	1275	65556	2827
FUEL "boolean"	1/74	31	65105	297930
STALL "boolean"	36/74	1681	50847	4904
DragTable* "boolean"	65/74	55	40377	1194
LOAD "boolean"	64/74	227	34899	1092
All model constraints "boolean"	0/74	6576	67046	≫ 336745
Two level	64/74	4307	34477	1205

Fig. 13. Performance of the various strategy combinations for the second mission.

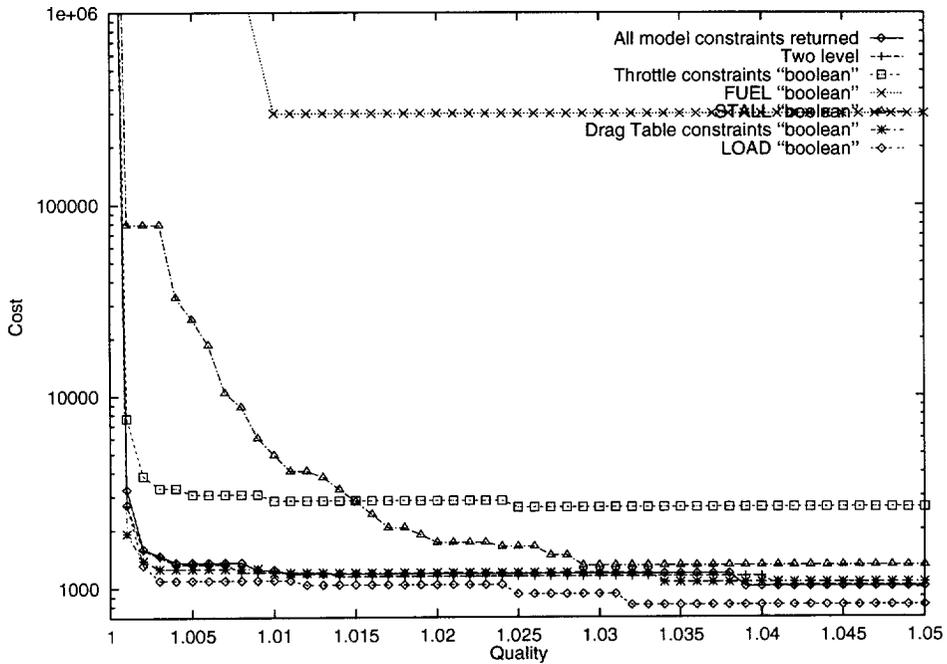


Fig. 14. Cost to achieve a range of design qualities with 99% confidence for the second mission. Quality is takeoff mass, normalized by the best takeoff mass found (Fig. 11), so quality = 1.01 corresponds to Fig. 13.

previous mission. A diagram of this best design appears in Fig. 12. Note that as this mission has a higher proportion of supersonic flight compared to the previous mission, the most efficient design for the mission as a whole has smaller, narrower wings than the best design for the previous mission. (An optimal design for a purely supersonic mission would have tiny wings and look like a missile.) We performed the same set of experiments for this case, and the experimental data which appears in Figs. 13, 14, 15, and 16 supports our previous conclusion that the model constraint communication strategy can cut search cost by an order of magnitude or more.

Strategy combination	Success	Start cost	Opt. cost	Est. 99% cost
All model constraints returned	48/74	41	57607	3429
Min/MaxThrottle "boolean"	13/74	5616	145770	48765
FUEL "boolean"	0/74	162	93146	≥ 426789
STALL "boolean"	39/74	7602	64000	5951
DragTable* "boolean"	34/74	342	131652	13352
LOAD "boolean"	51/74	1420	56171	3066
All model constraints "boolean"	0/74	133265	117224	≥ 1145732
Two level	49/74	231396	48049	16025

Fig. 15. Performance of the various strategy combinations for the second mission in the bigger box.

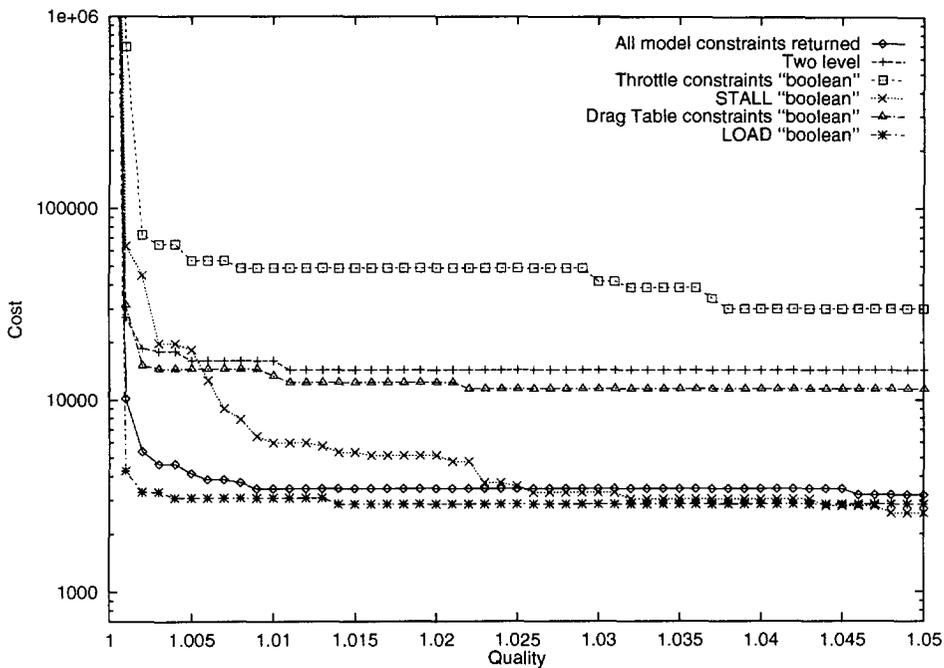


Fig. 16. Cost to achieve a range of design qualities with 99% confidence for the second mission in the bigger box. Quality is takeoff mass, normalized by the best takeoff mass found (Fig. 11), so quality = 1.01 corresponds to Fig. 15.

To get some independent validation of the quality of our designs, we presented our results to our domain expert, Dr. Gene Bouchard of Lockheed. Dr. Bouchard has extensive experience in traditional aircraft design and has also conducted research on the use of optimization in aircraft design, using both numerical optimization [2, 3] and genetic algorithms [4, 5]. Dr. Bouchard reported that model constraints appeared to provide significant improvements over optimization as used in industry today, particularly by allowing the use of large design spaces with very small feasible regions. As mentioned

previously, model constraints provide little benefit if expertise and historical experience are used to greatly restrict design variable ranges so that most of the design space is feasible. However, this restricted design space approach eliminates the possibility of finding better innovative designs in unexpected regions of the design space.

Dr. Bouchard examined the aircraft designs produced in our experiments and reported that his informal evaluation based on his experience was that our designs compared favorably with what he would expect from today's industry techniques, if applied with similar available domain knowledge and to similar mission specifications. However, a formal comparison of the particular numerical values generated by our design system to existing numbers from industry was not feasible. Designs seriously generated by industry use extensive proprietary information and design knowledge which would not be made available to academic researchers. For example, in Fig. 3 the weights computation would depend on proprietary information about how light various aircraft components could be, and the propulsion component would depend on very proprietary information about the engines' exact thrust capabilities in various flight regimes. Designing for our missions from scratch using standard industry techniques with the nonproprietary information available to us would have been a fairly involved project that was precluded both by funding considerations and by the level of aircraft industry collaboration available to us.

7. Related work

In [16] we examine the types of modeling knowledge that are needed so that a simulator can be reliably invoked by another program and we describe algorithms for detecting assumption violations and other problems that might lead to incorrect or unreliable simulation results. Strategies for communicating information about modeling failures to an automated design systems are not discussed, and the present paper is a sequel which remedies this deficiency.

The theme of our work in this paper is that existing "legacy" simulation software can in many cases be used in automated design with relatively minor enhancements. The alternative is to re-engineer a simulator declaratively, using an approach such as the compositional modeling introduced by [9] (which in turn builds on the qualitative process theory of [10]). Continuing along this research direction [11–13] discuss the use of qualitative simulation to check the correctness of numerical simulation results. By incorporating knowledge of potential model failures into the simulator, much of the need for communicating this information is alleviated. However, the cost of such re-engineering may be prohibitive in the case of large legacy simulation software.

Other automated intelligent controllers for numerical simulators are described in [14, 15, 30, 45, 48], but these do not address the issue of model and simulation quality assurance.

Intelligent monitoring for complex systems has received considerable attention (e.g., [7]), 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 [20, 25–27, 43], 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 [1–3, 6, 8, 19, 21, 28, 32, 33, 37, 42, 44], but have not addressed the issue of model and simulation quality assurance.

Work on the use of numerical optimization in aircraft design includes [22, 40, 41]. This work is to some extent able to avoid problems of model assumption violations by using human expertise to restrict design variable ranges to values which the model will tolerate. However, this approach is brittle in the sense that it will not scale to allow the sort of wide-ranging exploration of the design space which is possible with our explicit approach to modeling issues.

We have found that using model constraints results in good optimization performance in several other design domains. These domains include the design of racing yachts [8, 31–33, 37], the design of inlets for hypersonic jets [17, 38, 39], and the design of inlets for supersonic missiles [46, 47]. All of these domains use simulators that are more expensive than our aircraft simulator. For example, the simulators that we use for hypersonic inlets are computational fluid dynamics (CFD) codes that solve the two-dimensional Navier–Stokes equations, and take between 2 and 5 hours of CPU time for a single simulation. It would be very costly to perform extensive comparisons of different optimization strategies using such a simulator, so in those domains we have not performed the sorts of comparative studies we present in this article.

8. Limitations and future work

The model constraints communication strategy requires (of course) that model assumptions be representable using model constraint functions, which is a possible limitation since some problem domains may include assumptions which are not amenable to such representation. However, we did not encounter such a difficulty in the conceptual design of aircraft domain, and the appendix of this article presents another domain having significantly different sorts of model assumptions which are nevertheless also representable as model constraints. (Also, in Section 7 we list a number of other design domains in which we have successfully used model constraints.)

It appears that, for a given assumption in the aircraft domain, the design space can be partitioned into a small number of connected regions in which the assumption holds or does not hold. Under such conditions, even if a natural model constraint cannot be found, an artificial one may be constructible based on geometrical closeness to the region boundaries. However, if throughout the design space a model assumption oscillates with high frequency between satisfaction and violation, then it will be difficult to form a useful model constraint function. In this case the model will almost certainly need to be revised in order to be useful for design.

CFSQP, the numerical optimizer used in this article, assumes design variables take a continuous range of values, so our approach would not work as it stands for problems whose design variables are fundamentally limited to a discrete set of values. However,

the idea of representing model assumptions as constraints makes sense for discrete problems as well as for continuous problems, assuming appropriate search methods are used, such as methods from the extensive body of research on constraint satisfaction problems.

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 or Genetic Algorithms 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 a worthwhile area for future work. We have found model penalties in a Genetic Algorithm to be useful for the design of inlets for supersonic missiles [46,47]. (See also [29].)

An interesting but very challenging area for future research would be the automated generation of model constraint functions from a declarative representations of a model and its assumptions.

9. Conclusion

Automated search of a space of candidate designs is 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 assumption is violated. The search is then guided by using the values of these model constraint functions as constraint inputs to a standard constrained nonlinear optimization numerical method. A key result of our work is a successful demonstration of the application of AI techniques to an important engineering problem. In an empirical study of parametric conceptual aircraft design, we observed a cost improvement of two orders of magnitude. We also present evidence that such improvements may transfer to other domains. The principal contribution of our work is a new design optimization methodology which makes explicit the interaction between models of artifacts, and validity models of artifact models.

Acknowledgments

The research in this article depended critically on our collaboration with Gene Bouchard of Lockheed and Ron Luffy and Steve Scavo of General Electric Aircraft Engines. Our work benefited greatly from conversations with Saul Amarel, Tom Ellman, Haym Hirsh, Keith Miyake, Khaled Rasheed, Gerard Richter, Elisha Sacks, and Lou Steinberg. This research was partially supported by NASA under grant NAG2-817

and was 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.

Appendix A. Model constraints for a different design problem

We have also implemented model constraints for a different design problem having significantly different sorts of model assumptions. Here we look at the problem of designing a particular subcomponent of an aircraft: the engine's exhaust nozzle. In a supersonic aircraft, exhaust nozzles are complex, adjustable mechanical systems, and a computational simulation requires a number of model assumptions which have a geometrical flavor not found in the work we described above.

Fig. A.1 shows the class of nozzles we considered, the axisymmetric scheduled convergent-divergent exhaust nozzles often found in supersonic aircraft [24]. In Fig. A.1, r_{10} , r_e , and r_7 are fixed radii, and r_8 and r_9 are radii which are mechanically varied during aircraft operation. r_{10} is the outer radius of the engine to which the nozzle is attached, r_e is the radius of the duct leaving the engine, r_7 is the radius of the duct at the beginning of the movable convergent section of the nozzle, r_8 is the (variable) radius of the nozzle throat, and r_9 is the (variable) nozzle exit radius. Mechanically, this nozzle is a four-bar linkage, with three movable links labeled in Fig. A.1 by their lengths l_c , l_d , and l_e . During aircraft operation, the linkage is moved to change r_8 so that the cross-sectional area at the nozzle throat will produce desired engine performance. Since a four-bar linkage has one degree of freedom, setting r_8 also sets r_9 [18].

We approach the exhaust nozzle design problem as follows:

- (1) Design an aircraft for a particular mission as described in the main body of this article. A high level of abstraction is used, and the exhaust nozzle is not modeled explicitly.

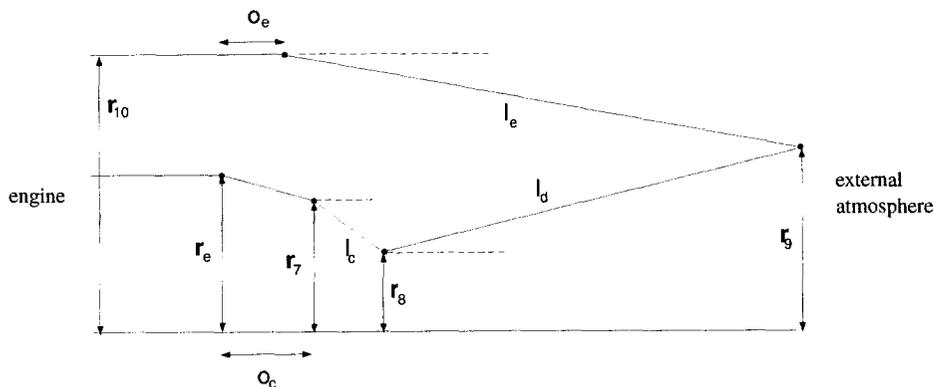


Fig. A.1. Axisymmetric convergent-divergent exhaust nozzle (flow from left to right).

- (2) Fix (almost) all of the major aircraft parameters at the optimal values found in the abstract optimization, add an explicit model of the exhaust nozzle to the simulation, and vary the parameters describing the nozzle geometry (Fig. A.1), using the same optimization criterion as in the main body of the article (i.e., minimize takeoff mass of the aircraft required to complete a given mission).

The explicit model of the exhaust nozzle used in the second stage is based on one-dimensional gas dynamics heavily supplemented by experimental data tables. The following additional model constraint functions are used (which, as before, are ≤ 0 if a constraint is satisfied and positive otherwise):

ELMAX = \langle length l_e of external nozzle flap $\rangle - \langle$ maximum length external nozzle flap could have, with the given values for the rest of the nozzle geometry, while still allowing the nozzle to be connected as a convergent-divergent nozzle \rangle .

ELMIN = \langle minimum length external nozzle flap could have, with the given values for the rest of the nozzle geometry, while still allowing the nozzle to be connected as a convergent-divergent nozzle $\rangle - \langle$ length l_e of external nozzle flap \rangle .

CA = \langle minimum angle to which convergent flap can move, while still maintaining a convergent-divergent configuration $\rangle - \langle$ maximum angle to which convergent flap can move, while still maintaining a convergent-divergent configuration \rangle .

R8LOW = \langle smallest value r_8 can achieve with current geometry $\rangle - \langle$ smallest value for r_8 required during mission simulation \rangle .

R8HIGH = \langle largest value for r_8 required during mission simulation $\rangle - \langle$ largest value r_8 can achieve with current geometry while maintaining a convergent-divergent configuration \rangle .

NG1 = $0 - z_7$. Nozzle geometry bound.

NG2 = $r_6 - r_{10}$. Nozzle geometry bound.

NG3 = $r_7 - r_{10}$. Nozzle geometry bound.

NG4 = $z_{10} - (z_7 + l_c + l_d)$. Nozzle geometry bound.

NG5 = $(r_7 - l_c) - r_6$. Nozzle geometry bound.

CA1LB, CA1UB, CA2LB, CA2UB: violation of bounds for a two-dimensional table of experimental data on nozzle angularity thrust loss.

CV1LB, CV1UB, CV2LB, CV2UB: violation of bounds for a two-dimensional table of experimental data on nozzle friction velocity/thrust loss.

CB1LB, CB1UB, CB2LB, CB2UB: violation of bounds for a two-dimensional table of experimental data on nozzle boattail (external) drag.

The model constraints defined in the main body of the article also continue to be used.

We experimentally compared the model constraints and boolean communication strategies for this new design problem, to see if the model constraints strategy again proved superior, as it did in the main body of this article. For our experiments, we focused on stage two of the two-stage design process described above, since stage one does not involve the explicit nozzle model. (Note that stage one is just the design process described extensively in the main body of the article.) In stage two we used a five-dimensional design space consisting of the four nozzle geometry parameters l_c , l_d , l_e , and r_7 , and

Strategy combination	Success	Start cost	Opt. cost	Est. 99% cost
All model constraints returned	5/100	21504	54297	68055
All model constraints "boolean"	3/100	2674928	39106	4103386

Fig. A.2. Performance of the various strategy combinations for the nozzle design problem.

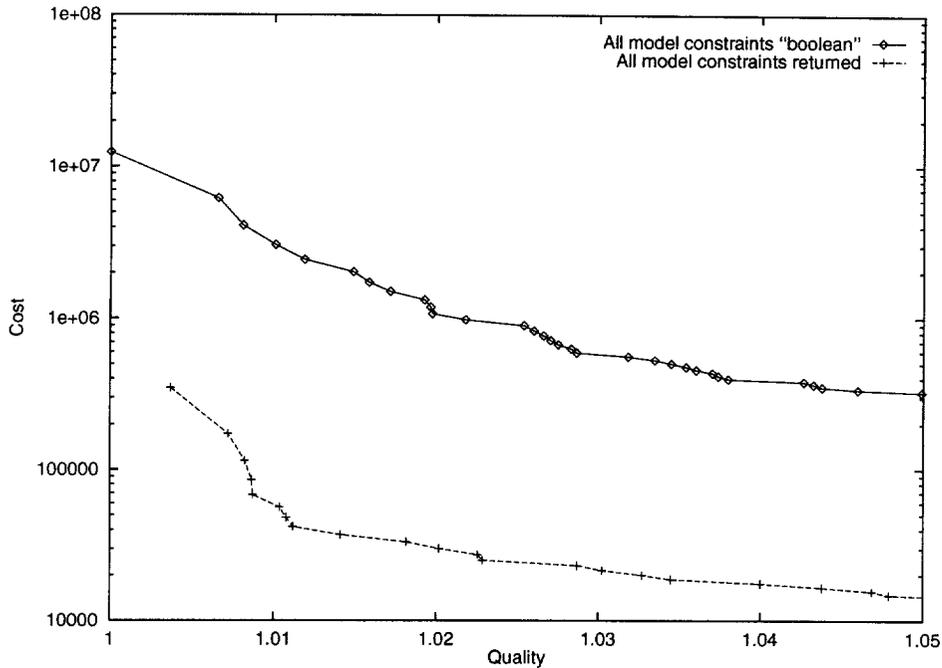


Fig. A.3. Cost to achieve a range of design qualities with 99% confidence for the nozzle design problem. Quality is takeoff mass, normalized by the best takeoff mass found.

also the aircraft wing area. Wing area was added to give some "looseness" to the main aircraft design, to avoid difficulties in finding a nozzle that will fit the exact optimal design found in stage one.

The mission of Fig. 2 was used again for these experiments. For the nozzle problem, we did not do extensive experiments testing each individual model constraint for its effect, but instead simply compared a multistart optimization in which all model constraints are available to CFSQP to an optimization in which all model constraints are handled with the "boolean" strategy. Figs. A.2 and A.3 show the experimental results, indicating that using model constraints reduces search cost by orders of magnitude, as we found in the main body of the article. Note that the density of feasible points in this space is only $100/2674928$ ($\approx 1/27000$), and that model constraints locate this small feasible region much more efficiently than do random probes.

As an additional experiment, after the second stage of the two-stage design process, we added a third stage in which a single optimization with all twelve design variables was started from the best point found in stage two. This third stage improved the design somewhat, but the improvement was rather small since the stage one and stage two design problems are fairly decomposable [34,36].

References

- [1] A.M. Agogino, A.S. Almgren, Techniques for integrating qualitative reasoning and symbolic computing, *Engineering Optimization* 12 (1987) 117–135.
- [2] E.E. Bouchard, Concepts for a future aircraft design environment, in: 1992 Aerospace Design Conference, Irvine, CA, 1992, AIAA-92-1188.
- [3] E.E. Bouchard, G.H. Kidwell, J.E. Rogan, The application of artificial intelligence technology to aeronautical system design, in: AIAA/AHS/ASSEE Aircraft Design Systems and Operations Meeting, Atlanta, GA, 1988, AIAA-88-4426.
- [4] M.F. Bramlette, E.E. Bouchard, Genetic algorithms in the parametric design of aircraft, in: L. Davis (Ed.), *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.
- [5] M.F. Bramlette, E.E. Bouchard, E. Buckman, L.A. Takacs, Current applications of genetic algorithms to aeronautical system, in: 6th Annual Aerospace Applications of Artificial Intelligence, 1990.
- [6] G. Cerbone, Machine learning in engineering: techniques to speed up numerical optimization, Ph.D. Thesis, Technical Report 92-30-09, Department of Computer Science, Oregon State University, Corvallis, OR, 1992.
- [7] D. Dvorak, B. Kuipers, Process monitoring and diagnosis, *IEEE Expert* 6 (3) (1991) 67–74.
- [8] T. Ellman, J. Keane, M. Schwabacher, Intelligent model selection for hillclimbing search in computer-aided design, in: *Proceedings 11th National Conference on Artificial Intelligence (AAAI-93)*, Washington, DC, MIT Press, Cambridge, MA, 1993, pp. 594–599.
- [9] B. Falkenhainer, K.D. Forbus, Compositional modeling: finding the right model for the job, *Artificial Intelligence* 51 (1991) 95–143.
- [10] K.D. Forbus, Qualitative process theory, *Artificial Intelligence* 24 (1984) 85–168.
- [11] K.D. Forbus, B. Falkenhainer, Self-explanatory simulations: an integration of qualitative and quantitative knowledge, in: *Proceedings 8th National Conference on Artificial Intelligence (AAAI-90)*, Boston, MA, 1990, pp. 380–387.
- [12] K.D. Forbus, B. Falkenhainer, Self-explanatory simulations: Scaling up to large models, in: *Proceedings 10th National Conference on Artificial Intelligence (AAAI-92)*, San Jose, CA, 1992.
- [13] K.D. Forbus, B. Falkenhainer, Scaling up self-explanatory simulations: polynomial-time compilation, in: *Proceedings 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, Montreal, Quebec, MIT Press, Cambridge, MA, 1995, pp. 1798–1805.
- [14] A. Gelsey, Using intelligently controlled simulation to predict a machine's long-term behavior, in: *Proceedings 9th National Conference on Artificial Intelligence (AAAI-92)*, Anaheim, CA, MIT Press, Cambridge, MA, 1991, pp. 880–887.
- [15] A. Gelsey, Automated reasoning about machines, *Artificial Intelligence* 74 (1) (1995) 1–53.
- [16] A. Gelsey, Intelligent automated quality control for computational simulation, *Artif. Intell. for Engineering Design, Analysis and Manufacturing* 9 (5) (1995) 387–400.
- [17] A. Gelsey, D.D. Knight, S. Gao, M. Schwabacher, NPARC simulation and redesign of the NASA P2 hypersonic inlet, in: 31st Joint Propulsion Conference, San Diego, CA, 1995, AIAA-95-2760.
- [18] A. Gelsey, D. Smith, Computational environment for exhaust nozzle design, *J. Aircraft* 33 (3) (1996) 470–476.
- [19] A. Gelsey, D. Smith, M. Schwabacher, K. Rasheed, K. Miyake, A search space toolkit: SST, *Decision Support Systems* 18 (1996) 341–356.
- [20] P.E. Gill, W. Murray, M.H. Wright, *Practical Optimization*, Academic Press, New York, 1981.
- [21] D. Hoeltzel, 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.

- [22] I. Kroo, S. Altus, R. Braun, P. Gage, I. Sobieski, Multidisciplinary optimization methods for aircraft preliminary design, in: 5th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Panama City, FL, 1994, AIAA-94-4325.
- [23] C. Lawrence, J. Zhou, 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, College Park, MD, 1995.
- [24] J.D. Mattingly, W.H. Heiser, D.H. Daley, *Aircraft Engine Design*, AIAA Education Series, American Institute of Aeronautics and Astronautics, New York, 1987.
- [25] J.J. Moré, S.J. Wright, *Optimization Software Guide*, SIAM, Philadelphia, PA, 1993.
- [26] P. Papalambros, J. Wilde, *Principles of Optimal Design*, Cambridge University Press, New York, 1988.
- [27] A.L. Peressini, F.E. Sullivan, J.J. Uhl Jr, *The Mathematics of Nonlinear Programming*, Springer, New York, 1988.
- [28] D. Powell, Inter-GEN: a hybrid approach to engineering design optimization, Ph.D. Thesis, Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, 1990.
- [29] K. Rasheed, H. Hirsh, A. Gelsey, A genetic algorithm for continuous design space search, *Artif. Intell. Engineering* 11 (3) (1997) 295–305.
- [30] E.P. Sacks, Automatic analysis of one-parameter ordinary differential equations by intelligent numeric simulation, *Artificial Intelligence* 48 (1) (1991) 27–56.
- [31] M. Schwabacher, The use of artificial intelligence to improve the numerical optimization of complex engineering designs, Ph.D. Thesis, Technical Report HPCD-TR-45, Department of Computer Science, Rutgers University, New Brunswick, NJ, 1996, <http://www.cs.rutgers.edu/~schwabac/thesis.html>.
- [32] M. Schwabacher, T. Ellman, H. Hirsh, Learning to set up numerical optimizations of engineering designs, *Artif. Intell. for Engineering Design, Analysis, and Manufacturing* 12 (2) (1998) (to appear).
- [33] M. Schwabacher, T. Ellman, H. Hirsh, G. Richter, Learning to choose a reformulation for numerical optimization of engineering designs, in: J. Gero, F. Sudweeks (Eds.), *Artificial Intelligence in Design '96*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1996, pp. 447–462.
- [34] M. Schwabacher, A. Gelsey, Multi-level simulation and numerical optimization of complex engineering designs, in: 6th AIAA/NASA/USAF Multidisciplinary Analysis & Optimization Symposium, Bellevue, WA, 1996, AIAA-96-4021.
- [35] M. Schwabacher, A. Gelsey, Intelligent gradient-based search of incompletely defined design spaces, *Artif. Intell. for Engineering Design, Analysis and Manufacturing* 11 (3) (1997) 199–210.
- [36] M. Schwabacher, A. Gelsey, Multi-level simulation and numerical optimization of complex engineering designs, *J. Aircraft* 35 (2) (1998) (to appear).
- [37] M. Schwabacher, H. Hirsh, T. Ellman, Learning prototype-selection rules for case-based iterative design, in: *Proceedings 10th IEEE Conference on Artificial Intelligence for Applications*, San Antonio, TX, 1994, pp. 56–62.
- [38] V. Shukla, A. Gelsey, M. Schwabacher, D. Smith, D.D. Knight, Automated redesign of the NASA P8 hypersonic inlet using numerical optimization, in: *AIAA Joint Propulsion Conference*, 1996.
- [39] V. Shukla, A. Gelsey, M. Schwabacher, D. Smith, D.D. Knight, Automated design optimization for the P2 and P8 hypersonic inlets, *AIAA J. Aircraft* 34 (2) (1997) 228–235.
- [40] J. Sobieszcanski-Sobieski, R.T. Haftka, Multidisciplinary aerospace design optimization: Survey of recent developments, in: 34th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, 1996, AIAA-96-0711.
- [41] J. Sobieszcanski-Sobieski, B.B. James, A.R. Dovi, Structural optimization by multilevel decomposition, *AIAA J.* 23 (11) (1985) 1775–1782.
- [42] S.S. Tong, D. Powell, S. Goel, Integration of artificial intelligence and numerical optimization techniques for the design of complex aerospace systems, in: *1992 Aerospace Design Conference*, Irvine, CA, 1992, AIAA-92-1189.
- [43] G.N. Vanderplaats, *Numerical Optimization Techniques for Engineering Design: With Applications*, McGraw-Hill, New York, 1984.
- [44] B.C. Williams, J. Cagan, Activity analysis: the qualitative analysis of stationary points for optimal reasoning, in: *Proceedings 12th National Conference on Artificial Intelligence (AAAI-94)*, Seattle, WA, 1994, pp. 1217–1223.

- [45] K.M.-K. Yip, Understanding complex dynamics by visual and symbolic reasoning, *Artificial Intelligence* 51 (1–3) (1991) 179–221
- [46] G.-C. Zha, D. Smith, M. Schwabacher, K. Rasheed, A. Gelsey, D. Knight, M. Haas, High performance supersonic missile inlet design using automated optimization, in: 6th AIAA/NASA/USAF Multidisciplinary Analysis & Optimization Symposium, Bellevue, WA, 1996, AIAA-96-4142.
- [47] G.-C. Zha, D. Smith, M. Schwabacher, K. Rasheed, A. Gelsey, D. Knight, M. Haas, High performance supersonic missile inlet design using automated optimization, *AIAA J. Aircraft* 34 (6) (1997) 697–705.
- [48] F. Zhao, Extracting and representing qualitative behaviors of complex systems in phase space, *Artificial Intelligence* 69 (1–2) (1994) 51–92.