A Search Space Toolkit

Andrew Gelsey Don Smith Mark Schwabacher

Khaled Mohamed Rasheed Shehata Keith Miyake

Computer Science Department

Rutgers University

New Brunswick, NJ 08903

USA

(908) 445-4869 (FAX -5691)

{gelsey,dsmith,schwabac,shehata,kmiyake}@cs.rutgers.edu

Abstract

The Search Space Toolkit (SST) is a suite of tools for investigating the properties of the continuous search spaces which arise in designing complex engineering artifacts whose evaluation requires significant computation by a numerical simulator. SST has been developed as part of NDA, a computational environment for (semi-)automated design of jet engine exhaust nozzles for supersonic aircraft which resulted from a collaboration between computer scientists at Rutgers University and design engineers at General Electric and Lockheed. Though the design spaces for this sort of engineering artifact are mainly continuous, they typically include features such as unevaluable points, multiple local optima, and large derivatives which cause difficulties for standard numerical optimization methods. The search spaces which SST explores also differ significantly from the discrete search spaces that typically arise in artificial intelligence research, and properly searching such spaces requires a synergistic combination of numerical methods and AI techniques and is a fundamental AI research area. By promoting the design space to be a first class entity, rather than a "black box" buried in the interface between an (unconstrained) optimizer and a simulator, SST allows a more principled approach to automated design.

To appear in *Decision Support Systems, special issue on Unification of Artificial Intelligence with Optimization*, 1996.

1 Introduction

Various researchers have addressed decision making problems with unified approaches which merge optimization techniques with artificial intelligence techniques. Approaches using optimization in conjunction with subfields of AI including case-based reasoning [Lee and Kim 1993] and other machine learning [Schwabacher *et al.* 1994, Schwabacher *et al.* 1995], constraint satisfaction problems [Lee and Kim 1995], and rule-based systems [Lee and Song 1995] have been explored. In the present article we extend the research on unified AI/optimization approaches, focusing on the combination of optimization with ideas from two subfields of AI: search and problem formation.

The Search Space Toolkit (SST) is a suite of tools for investigating the properties of continuous search spaces. The search spaces which SST explores differ significantly from the discrete search spaces that typically arise in artificial intelligence research, and properly searching such spaces is a fundamental AI research area. Our SST research has focused on the problem of designing complex engineering artifacts and the analysis of the associated search spaces. Evaluation of points within these search spaces requires significant computation by a numerical simulator. The knowledge of search space properties which SST extracts can be used as a basis for *AI-augmented optimization*, in which numerical optimization algorithms are supplemented by AI techniques to improve their ability to find optima in complex, realistic search spaces.

SST has been primarily developed as part of the Nozzle Design Associate (NDA) project [Gelsey and Smith 1995]. NDA is a computational environment for (semi-)automated design of jet engine exhaust nozzles for supersonic aircraft. NDA was developed in a collaboration between computer scientists at Rutgers University and design engineers at General Electric and Lockheed. The NDA project has two principal goals: to provide a useful engineering tool for exhaust nozzle design, and to explore fundamental research issues that arise in the application of automated design optimization methods to realistic engineering problems.

Figure 1 shows the class of nozzles supported by the current NDA, the axisymmetric scheduled convergent-divergent exhaust nozzles often found in supersonic aircraft. [Mattingly

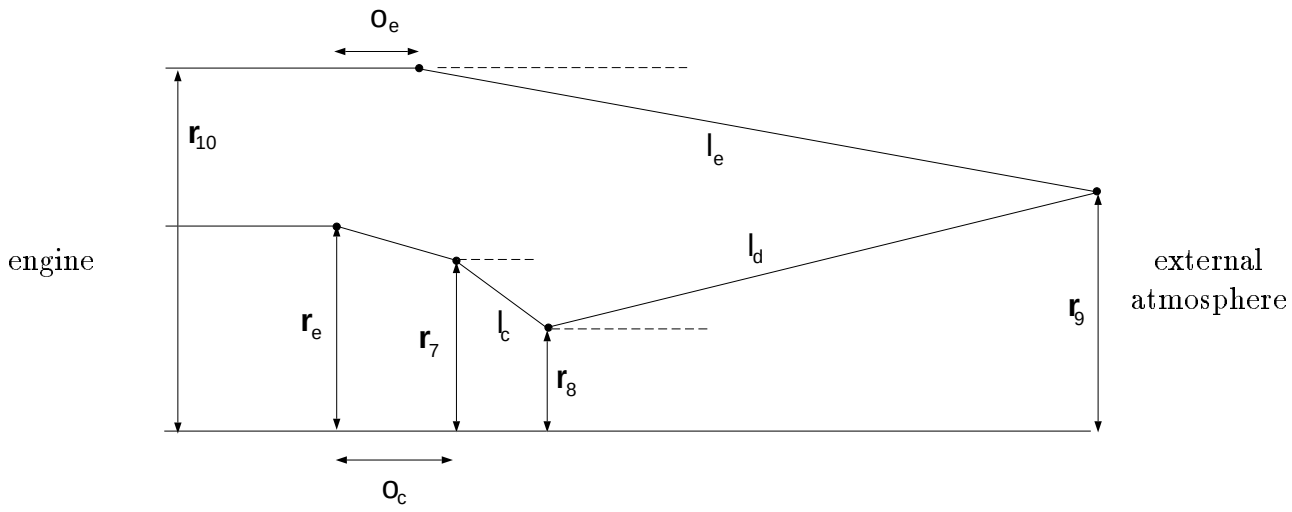


Figure 1: Axisymmetric convergent-divergent exhaust nozzle (flow from left to right)

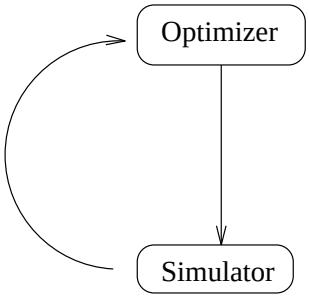


Figure 2: Simple simulate/modify loop

et al. 1987] In Figure 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 Figure 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. The job of NDA is to choose values for the parameters l_c, l_d, and l_e that give optimal performance for a particular aircraft and flight mission.

Figure 2 shows what might be called a "naive" approach to design automation: simply

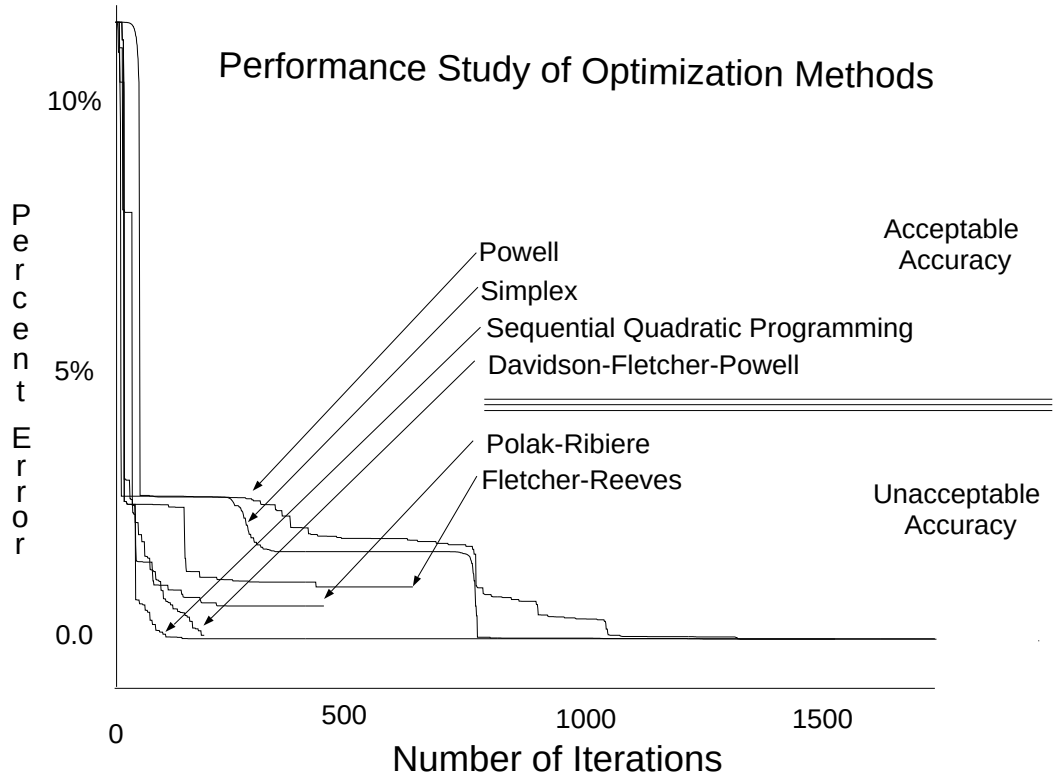


Figure 3: Experimental data showing quality of termination points of various optimization methods

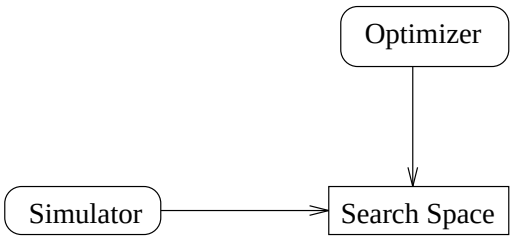


Figure 4: Optimizer searches space induced by simulator

combine a standard optimizer with a simulator capable of evaluating candidate designs. Unfortunately, simulators are typically written with the assumption that they will be invoked by experienced human users, and making them robust enough for use in an automated environment like Figure 2 can be demanding. Even when some software engineering has been done to make the simulator and optimizer capable of working together, optimization results tend to vary widely, as illustrated by the example in Figure 3, in which an exhaust nozzle simulator is combined with a number of different optimization algorithms from [Press *et al.* 1992] and [Lawrence *et al.* 1995]. In Figure 3, each optimizer was started at the same point and run until it could find no further design improvement. The vertical axis in Figure 3 shows the deviation of the design quality of each point found by an optimizer from the best point found by any method, and the horizontal axis shows the number of iterations used to reach each point. In Figure 3, we sort the optimization methods into groups: those whose deviation was small enough to be "acceptable" for the current design goals, and those with larger, unacceptable deviation.

Figure 4 illustrates an alternative way of looking at the problem of automated design optimization. The viewpoint here is that the simulator implicitly defines a search space, which in turn is searched by the optimizer. The premise of our SST research is that automated design optimization has a much better chance of success if this search space is treated as a distinct entity whose geometry and topology should be investigated by a variety of computational tools, rather than as a "black box" buried in the interface between an optimizer and a simulator.

In the current version of NDA, the design parameters defining the search space are the

lengths of the convergent, divergent, and external nozzle flaps (l_c, l_d, and l_e in Figure 1). NDA optimizes the nozzle design under the constraint that the aircraft must be able to complete its designated mission, and with the goal that cost should be minimized. NDA currently uses gross takeoff mass as an approximation for cost, as takeoff mass is a rough combination of both acquisition cost (approximated by dry mass) and operating cost (approximated by fuel mass). To compute gross takeoff mass, NDA uses a detailed numerical simulation of the relevant physics. [Gelsey and Smith 1995]

Section 3 of this article describes search space properties of interest for design automation, Section 4 describes the tools in SST for investigating search space properties, and Section 5 discusses our experiments in "AI-augmented optimization", in which knowledge of the design search space acquired by the search space toolkit is used to improve the ability of numerical optimizers to find an optimal design.

2 Related Work

[Lee and Kim 1995] describes UNIK-OPT, a knowledge-assisted optimization model formulation system. [Lee and Kim 1993] describes UNIK-CASE, which uses case-based methods to provide knowledge for use by UNIK-OPT. [Lee and Song 1995] describes a framework which unifies an optimization model with a rule-based system.

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. Search has been a key focus of AI research from the field's beginning [Charniak and McDermott 1987], but most of the attention has been on discrete rather than continuous objective functions. A number of research efforts have combined AI techniques with numerical optimization [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,

- number of local optima
- convexity
- "depth" of local optima
- smoothness; continuity of n^{th} derivative
- local properties in piecewise smooth regions
- evaluability of objective function
- topology/geometry of evaluable region
- constraints: explicit, implicit
- ridges; valleys
- plateaus

Figure 5: Search space properties

Hoeltzel and Chieng 1987, Cerbone 1992], but automated identification of search space properties has not been a focus in this work.

Other work in our Rutgers AI/Design research group used machine learning [Weiss and Kulikowski 1991] to improve optimization by generating abstractions and decompositions of the search space [Ellman and Schwabacher 1993], by selecting a prototype defining an appropriate search space for a given design goal [Schwabacher *et al.* 1994], and by reformulating the search space through the incorporation of constraints into the design modification operators [Schwabacher *et al.* 1995].

3 Search Space Properties

Figure 5 lists a number of search space properties that are likely to be important in searching the space for an acceptable design. One extremely important property of a search space is the number of local optima. Search techniques that work well on a space with a single local optimum, which is therefore the global optimum, may be ineffective for spaces with multiple local optima. Which search techniques are most appropriate will depend greatly on the number of local optima. If it is possible to show that the objective function is "convex" (Hessian matrix positive-definite everywhere) and that the feasible region is convex (geometric sense) then the function can have at most one one local optimum. [Vanderplaats

1984] An associated property is the "depth" of the local optima, i.e., how much worse the objective function gets before leaving the basin of attraction of a local optimum.

Numerical optimization algorithms [Vanderplaats 1984] tend to be quite sensitive to the smoothness of the objective function. SST focuses on problems in which the objective function to be optimized by the search process is computed by a complex numerical simulation of the physics of the artifact being designed. Such an objective function is often not very smooth. Continuity of the objective function itself is important for almost any numerical optimizer, and degree of smoothness (i.e., continuity of the first, second, or even higher derivatives) is important for several optimization methods.

As we will discuss in the following section, SST builds a global picture of a search space by combining information gathered at different local points. For example, if a point is suspected of being a local optimum, information about derivatives of the objective function should be gathered. If the objective function is not smooth at the point under consideration, then the neighborhood about the point should be divided into piecewise smooth regions so derivatives can be computed (numerically) in each region. The gradient (vector of first partial derivatives of the objective function with respect to the design parameters) is an important piece of data, since if it is zero that often indicates that a point is a local optimum. The Hessian (matrix of second partial derivatives of the objective function with respect to pairs of design parameters) is also useful, and if the Hessian is diagonalized, its eigenvalues and eigenvectors may indicate important directions in the search space.

An issue of great practical importance that tends to be ignored by AI work on search and by work in numerical optimization is the evaluability of the objective function. In designing a complex engineering artifact, the objective function is computed by numerical simulation. The numerical simulator is in effect "executing" a model of the physical system being designed, and like any model of physics this model must be based on various approximations and simplifying assumptions. However, it is quite likely that some combinations of values for the design parameters will result in physical configurations that violate the assumptions upon which the model is based, thus making the objective function unevaluable at that point

of the search space.

One obvious "fix" for an objective function which is not evaluable everywhere is to extend the simulator to use a more complete model which is consistent with the "bad" set of parameters. However, this effort typically cannot be justified unless it is likely that the unevaluable regions contain good designs. Often the good designs themselves will in fact satisfy the model assumptions. For example, certain sets of parameter values for the exhaust nozzle in Figure 1 will make it impossible to bring the exhaust to a supersonic velocity under certain flight conditions. In this case, the nozzle will have a very low thrust and thus be a bad design. However, the basic nozzle model depends on the assumption that flow will become supersonic in part of the nozzle, so this set of nozzle parameters will actually not be evaluable by the simulator. The problem is that, while this design is not a good one, it may be encountered by an optimizer as it searches the design space, and since it is unevaluable, the optimizer will have no information to guide it towards better regions of the search space.

Thus the topology and geometry of the evaluable region[1] of the search space are important properties. A basic topological question is whether the evaluable region is connected. If the evaluable region has several disconnected components, then it is quite unlikely that a search started in one component will ever make its way into a different component. If it turns out that the evaluable region is connected, then a second question is whether it is simply connected, or if there are "holes" — pockets of unevaluable points inside the evaluable region. Geometrical properties may also be important: for example, whether the evaluable region is a convex geometric shape.

Constraints on the allowable set of solutions within the search space are also an important property. Explicit constraints will typically be clear from the problem statement, but as the discussion above indicates, there may also be implicit constraints such as not violating modeling assumptions. By an "implicit constraint", we mean a constraint which is satisfied at design points where the simulator is able to finish its computation and return a value for

[1]The evaluable region is the set of design points at which the simulator's assumptions are satisfied so that the objective function can be correctly computed.

the objective function, and which fails to be satisfied at design points where the simulator "crashes" and cannot compute a value for the objective function. Ideally, simulators should never crash, but "real-world" engineering simulators often do, unfortunately.

Ridges, valleys ("negative ridges"), and plateaus are also important search space features. Ridges and valleys are regions in which movement in a particular direction causes the objective function to change very rapidly, potentially masking weaker changes in an orthogonal direction which may nevertheless eventually lead to improvement of the objective function. Plateaus are regions where the objective function changes very slowly — they may trap optimizers.

4 Tools

The organization of SST follows a "toolkit" approach rather than an "environment" approach: different capabilities are implemented in different "tools" (programs) which the user runs from an operating system "shell", rather than from a special SST environment. Some SST tools also internally call other SST tools.

As mentioned in the previous section, the number of local optima is a critical property. Unfortunately, for an objective function defined by a large numerical simulation program, the information we are able to obtain about the number of local optima will generally be statistical in nature, rather than the subject of a mathematical proof. SST uses a Monte-Carlo-like multistart method for estimating the number of local optima: the algorithm repeatedly chooses random combinations of design parameters, uses the resulting design as a starting point for a numerical optimizer, and sorts the termination points of the optimizations into bins. (A byproduct of this process may be the identification of a global optimum, which is the best of the local optima.)

Of the nine numerical optimizers we have tried, the best performance in terms of speed and reliability was produced by CFSQP, a state-of-the-art implementation of the Sequential Quadratic Programming method [Lawrence *et al.* 1995]. Sequential Quadratic Programming

start point	takeoff mass	l_c	l_d	l_e	function evaluations	local optimum
1	306525.8	4.4	37.5	57.4	299	A
2	306526.9	4.3	37.2	57.0	387	A
3	306527.2	4.3	37.0	56.8	231	A
4	306524.7	4.3	37.3	57.1	351	A
5	306995.1	2.6	35.2	52.7	371	B
6	306526.6	4.4	37.7	57.7	355	A
7	306995.1	2.6	35.2	52.7	475	B
8	306524.9	4.4	37.3	57.2	261	A
9	306995.1	2.6	35.2	52.7	517	B
10	306525.0	4.3	37.4	57.3	305	A

Figure 6: Multistart using CFSQP to minimize aircraft takeoff mass

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. Figure 6 shows results of a multistart using CFSQP in the NDA nozzle design search space. For ten optimization runs all with different randomly generated starting points, Figure 6 shows the takeoff mass (in kilograms) and nozzle flap lengths (in inches) of the best nozzle design found by the optimizer with each starting point. (As described in Section 1, the design goal is to minimize takeoff mass while satisfying mission requirements.) CFSQP terminated when it could find no further local improvement, and Figure 6 shows how many function evaluations (i.e., mission simulations) occurred during the optimization. Note that the space appears to have two local optima.

SST currently addresses the issue of objective function evaluability by a fixed grid sampling technique, both on large regions and on selected subregions of a search space. In the NDA exhaust nozzle search space, if SST imposes a grid on a large section of the search space spanning the "reasonable" range of values for the design parameters, sampling the grid points reveals that only about 4% of the grid points are evaluable. These grid points

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

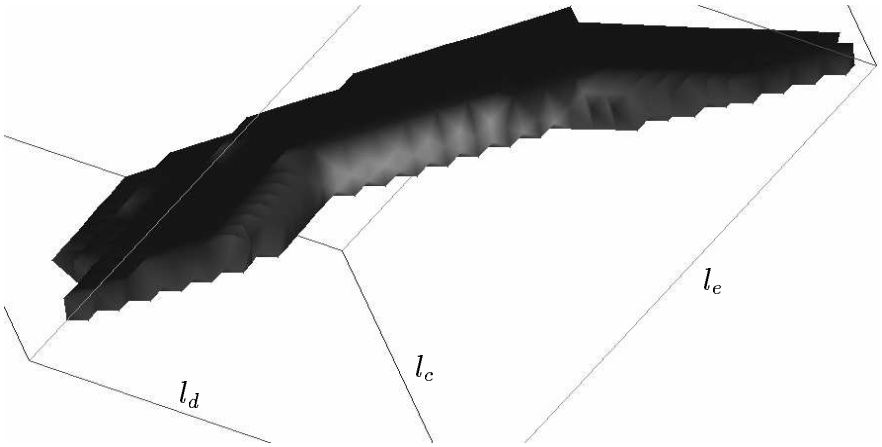


Figure 7: The "slab-shaped" evaluable region (We normally display this in color on a workstation screen, with differing colors indicating differing design quality.)

are contiguous, and form a "slab-shaped" evaluable region. Figure 7 graphically portrays the appearance of this slab using the AVS[3] scientific visualization software. In the current NDA, the space of possible nozzle designs is three dimensional, since we only allow NDA to vary the three parameters l_c, l_d, and l_e, the lengths of the movable nozzle flaps. SST does not detect internal pockets of unevaluable points within this slab, suggesting that the evaluable region in this space is simply connected. The boundaries of this slab are implicit constraints on the acceptable combinations of design parameters for this problem. For the purpose of performing design optimization, when computing partial derivatives the NDA dynamically modifies the derivative step size and direction in order to avoid the unevaluable region, and a software wrapper returns to the optimizer a large "bad" value for unevaluable points that are encountered outside of gradient computation (i.e., a large nonsmooth penalty function is applied at unevaluable points).

Several of the SST tools have led us to the conclusion that the gross structure of the NDA nozzle design space is that of a valley. The SST fixed grid sampling reveals that the slab-like evaluable region has a thin surface running midway between the flat boundaries of the slab which contains designs much better than their neighbors closer to the outside of the

[3]Advanced Visualization System

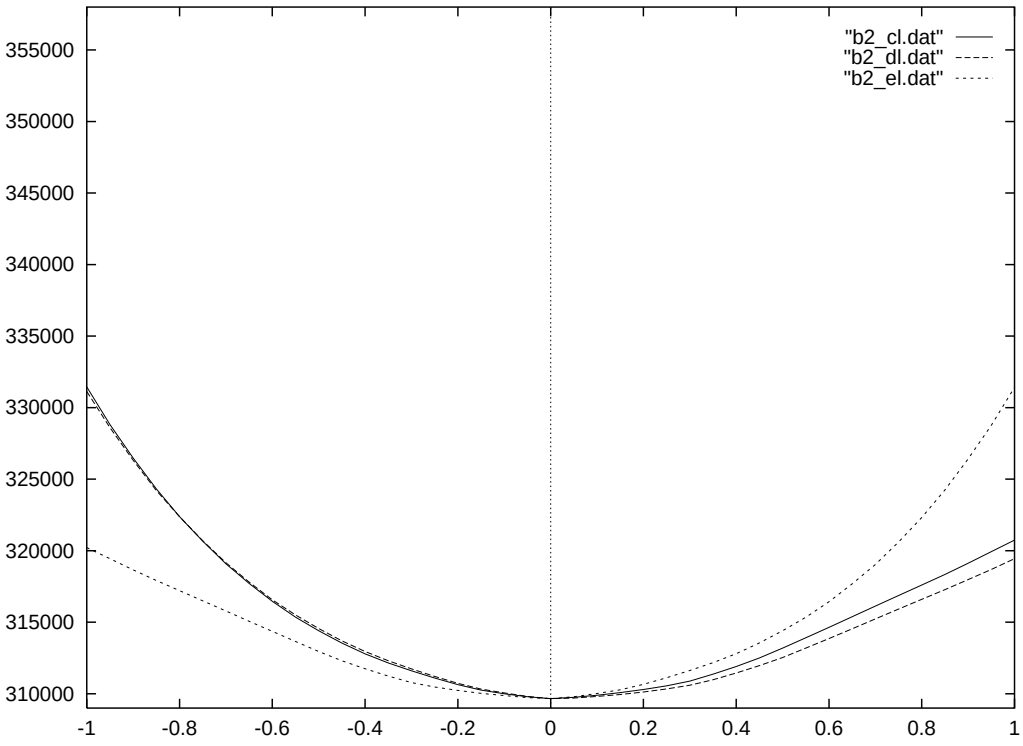
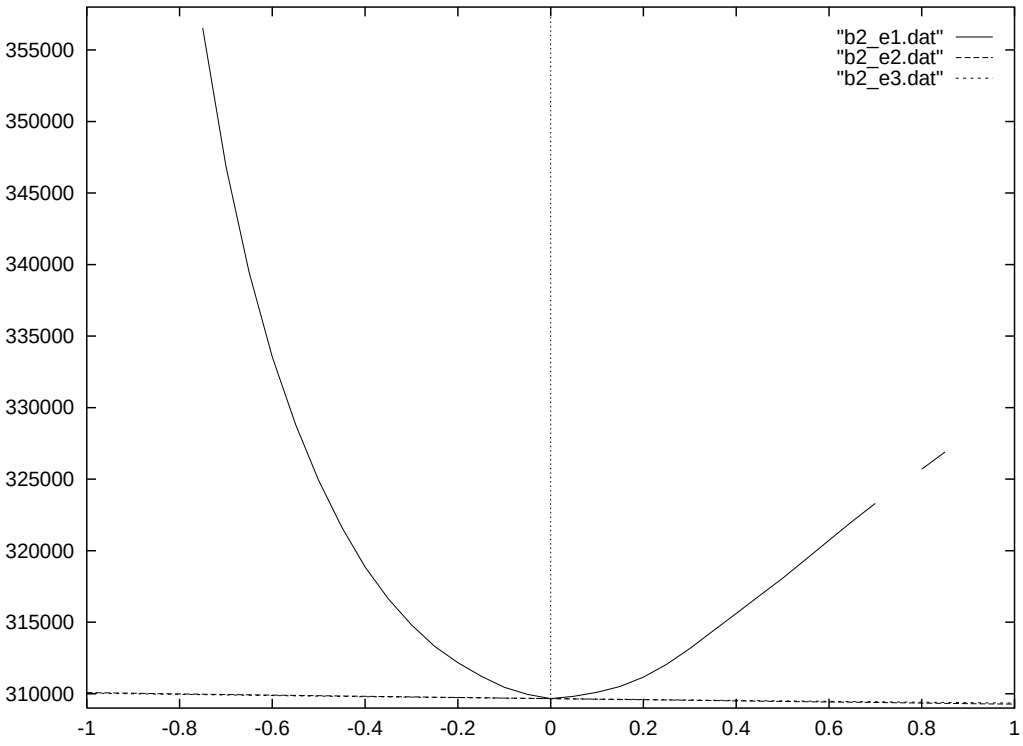


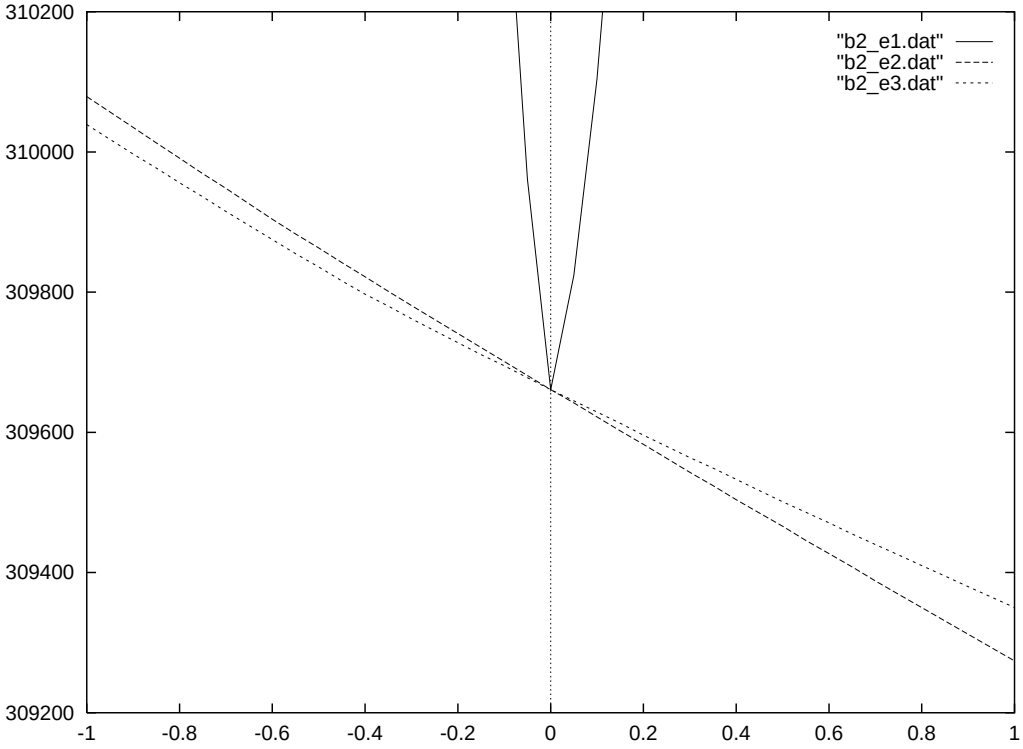
Figure 8: Objective as function of Δl_c, Δl_d, and Δl_e

slab. The optimizations run by the Monte-Carlo-like multistart techniques tend to stop on this central surface, though many of the stopping points are not true local minima. If the Hessian matrix for a point on this central surface is diagonalized, one eigenvalue is much larger than the others, and its corresponding eigenvector is normal to the central surface. This data gathered by SST strongly suggests that the central surface running through the middle of the slab is a higher dimensional analog of a ridge. The nozzle design objective function is an approximation of cost, which should be minimized, so we refer to this ridge as a "valley" and we refer to the central surface in the slab as the "valley floor". Optimizers tend to stop soon after finding the valley floor because the gradients driving the optimization towards the valley floor are very strong and tend to mask the much weaker gradients along the valley floor.

Figure 8 shows how the objective function varies about a "typical" optimization stopping point as a function of change in design space parameters (l_c, l_d, and l_e in Figure 1), and



a) Large scale



b) Close-up view

Figure 9: Objective function variation in directions of Hessian eigenvalues

indicates that the gradient is approximately zero and the second derivatives positive, so that it was "legitimate" for the optimizer to stop here. Figure 9 shows how the objective function varies about the same optimization stopping point as a function of combinations of parameters in the direction of the eigenvectors of the Hessian at this point. Here we see that there may in fact be a "downhill" direction which is a linear combination of the eigenvectors corresponding to the two smaller eigenvalues, but that the other much larger eigenvalue is "masking" this possibility for improvement.

There is more to a search space than its gross structure. If gross analysis reveals that a search space is a valley, the next natural question is "what is the structure of the valley floor?". To investigate this issue, SST includes a tool we call "dimension reduction". The floor of a valley can be considered a search space in its own right, but a search space of dimensionality one less than that of the primary search space. Though the valley floor space has fewer dimensions, it may still have a very complex structure. To ascertain the properties of this subspace without having them masked by the strong gradients in the rest of the primary search space, SST must limit its evaluations to points exactly on the valley floor.

The SST dimension reduction algorithm works by projecting the desired subspace onto a hyperplane tangent to the subspace at some point. (A limitation of our current version of this algorithm is that it works poorly for subspaces with high curvature.) Linear algebra gives a coordinate system for the hyperplane with one less dimension than the primary space. This coordinate system then serves as a coordinate system for the subspace by identifying each point P_s in the subspace with the nearest point P_h on the hyperplane. (I.e., the line defined by P_s and P_h is normal to the hyperplane.) Thus each function evaluation in the reduced dimension subspace requires a search in the primary space along the line normal to the corresponding point on the hyperplane in order to find the intersection with the subspace and evaluate the point of intersection. If the subspace is the valley floor in the nozzle search space, then each function evaluation requires solving a one-dimensional minimization problem, because the line normal to the hyperplane (just a plane in this case) will have its minimum value of the objective function where it intersects the valley floor.

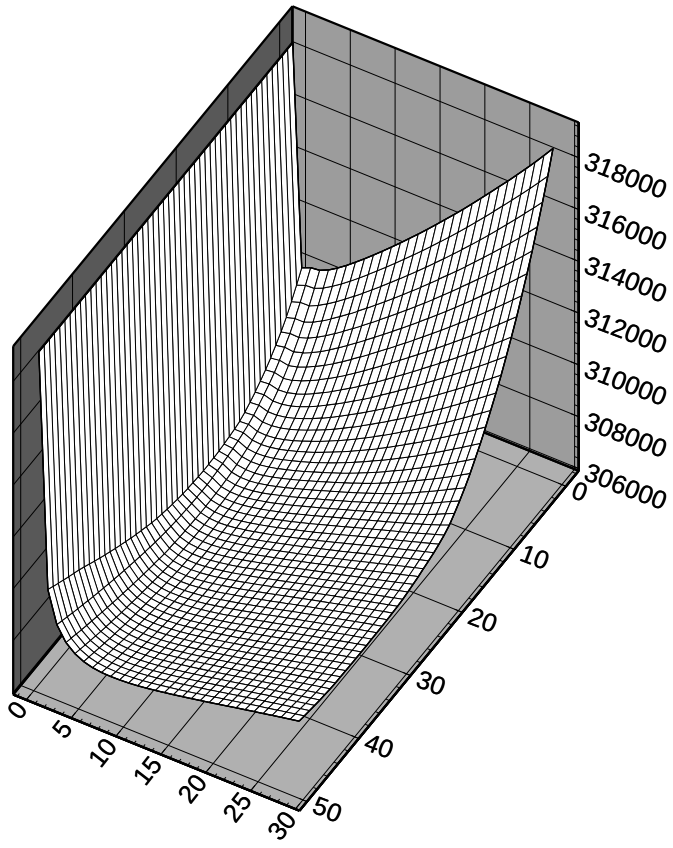


Figure 10: Valley floor structure for NDA nozzle design space (one dimension less than full nozzle design space)

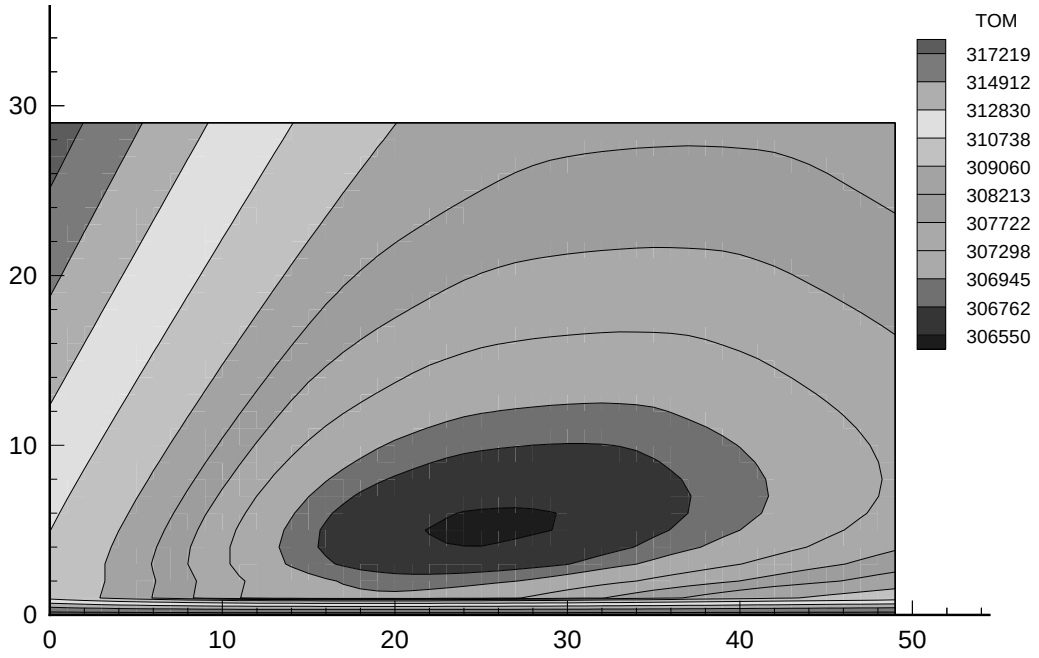


Figure 11: Valley floor contour plot

The SST dimension reduction algorithm has been applied to the NDA nozzle design search space. By combining the dimension reduction algorithm with our fixed grid sampling technique, we were able to determine the structure of the valley floor for the nozzle design space. Figure 10 shows the reduced dimension nozzle design space computed by this algorithm. This "valley floor" space has one less dimension than the full 3D nozzle design space. (Figure 11 shows a contour plot of the same data).

Figures 12 and 13 show "closeups" of the valley floor near the global optimum. Here a finer fixed grid sampling of the dimension-reduced space has been used.[4] Note that the range of takeoff masses (vertical axis in Figure 12) does not range as high as in Figures 10 and 11. Also note the apparent presence of two local optima, which is particularly clear in Figure 12. The locations of these optima are the same (to within grid spacing) as the locations of the two local optima found by the multistart of Figure 6, providing an internal consistency check between the two methods.

[4]Note: the axes for Figures 12 and 13 are not labeled with the same units as the axes for Figures 10 and 11

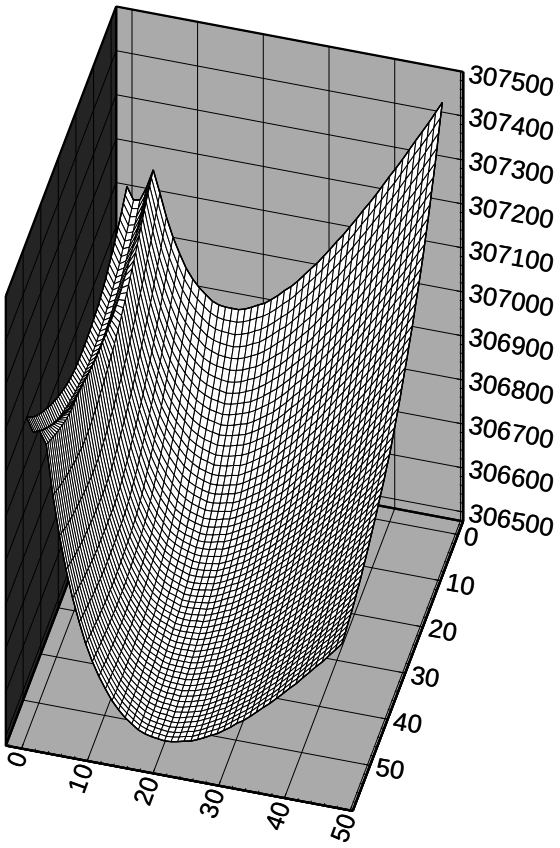


Figure 12: Valley floor structure (closeup view)

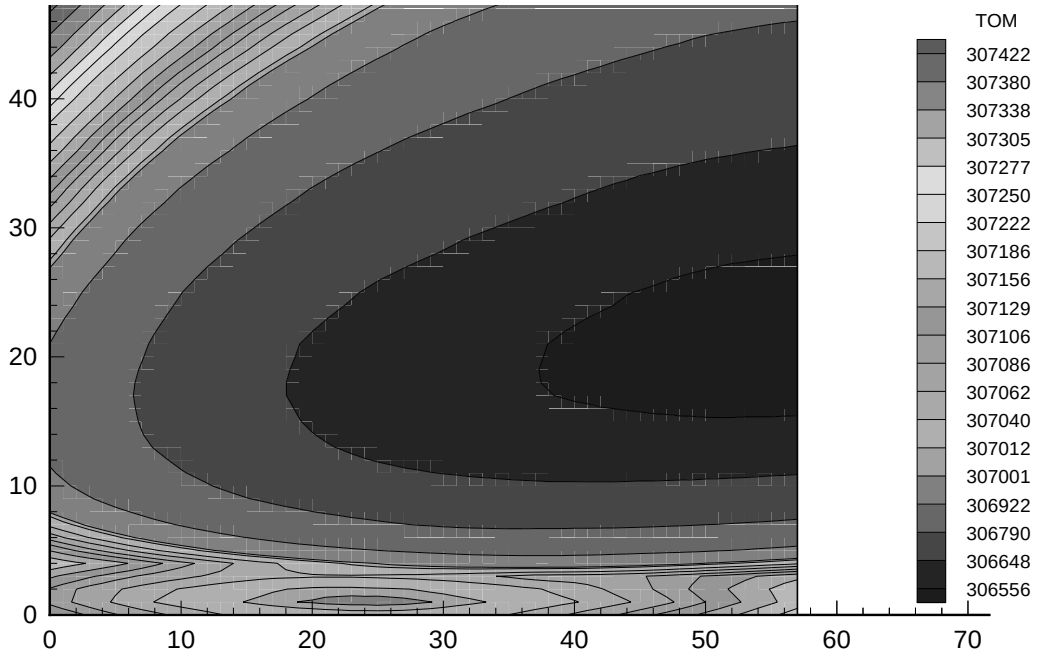


Figure 13: Valley floor contour plot (closeup view)

The use of scientific visualization software to display an entire search space is only possible for a three-dimensional search space, as in the visualization shown in Figure 7. However, extensions of the search space toolkit dimension reduction algorithms may be useful for extracting two- or three-dimensional "slices" of higher dimensional search spaces which can then be displayed using visualization techniques for 3D spaces (as used in Figure 7) or 2D spaces (as used in Figures 10 through 13).

5 AI-augmented Optimization

The search space properties which SST extracts can be used as a basis for *AI-augmented optimization*, in which numerical optimization algorithms are supplemented by AI techniques to improve their ability to find optima in complex, realistic search spaces. A particular form of AI-augmented optimization is the use of problem reformulation to improve optimizer performance. Problem reformulation has long been an important area of theoretical AI research — here we demonstrate that problem reformulation can have practical benefits as

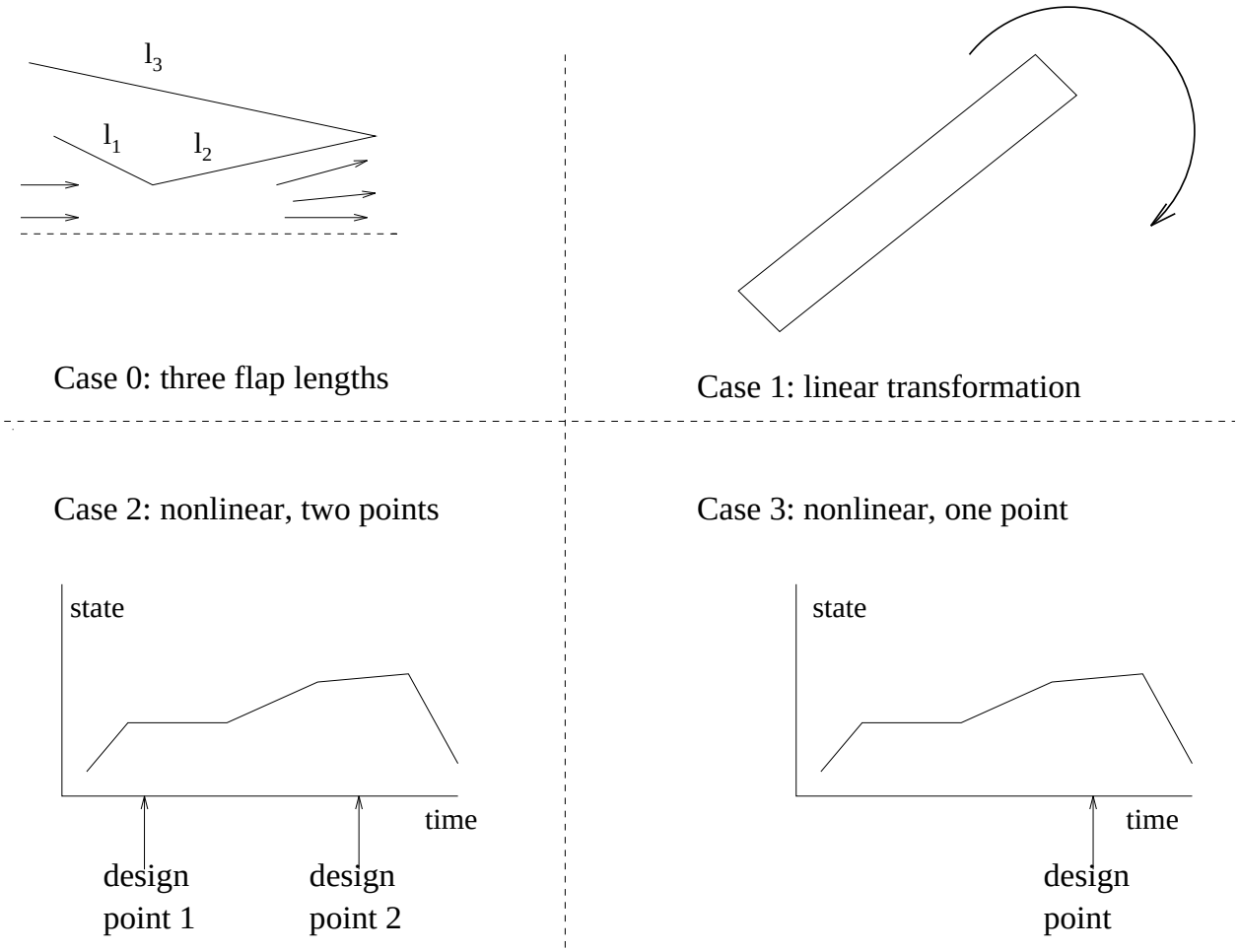


Figure 14: Reformulations of the exhaust nozzle design problem

well.

Several possible problem reformulations are suggested by the characteristics of the NDA nozzle design search space which SST identifies. Figure 14 shows some problem reformulations we have investigated. In each case, the reformulated problem involves three design parameters. In case 0 of Figure 14, which is the original nozzle design problem described in Section 1, the three parameters are the lengths of the three movable nozzle flaps. The other three cases involve transformed sets of parameters which are described below.

We implement each particular problem reformulation by means of a software mediator which we call a "wrapper", as shown in Figure 15. To external agents such as optimizers

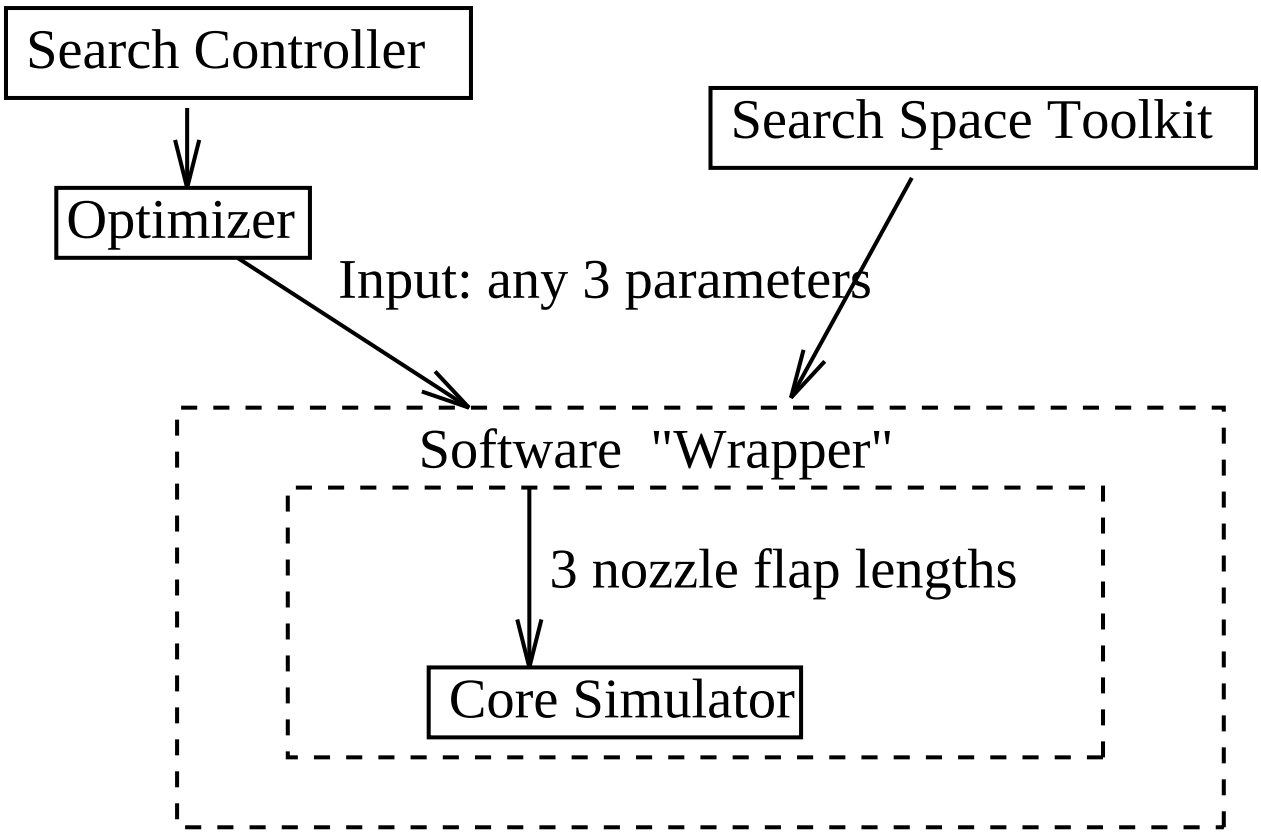


Figure 15: Wrappers for exhaust nozzle design problem

start point	takeoff mass	p_1	p_2	p_3	function evaluations	local optimum
1	306524.7	-9.0	43.4	51.9	274	A
2	306524.9	-9.0	43.6	52.2	203	A
3	306525.0	-9.0	43.3	51.7	213	A
4	306995.2	-8.6	39.1	49.2	228	B
5	306995.1	-8.7	39.1	49.2	233	B
6	306524.9	-9.0	43.5	52.1	210	A
7	306524.8	-9.0	43.4	51.9	384	A
8	306524.7	-9.0	43.5	52.0	214	A
9	306524.9	-9.0	43.5	52.1	165	A
10	306524.7	-9.0	43.4	51.8	591	A

Figure 16: Multistart CFSQP in rotated space

or the SST, the wrapper "looks" just like the core nozzle simulator — they both accept three design parameters and return a single measure of merit. To the core nozzle simulator inside, the wrapper "looks" just like the external agents that typically invoke it. Thus we can experiment with any reparameterization of the nozzle design problem without having to change any software other than the wrapper itself.

Both the very large-scale information in Figure 7 and the very small-scale information in Figures 8 and 9 indicate the importance of the spatial direction normal to the slab. One natural problem reformulation is case 1 in Figure 14, a linear transformation which rotates the coordinates of the search space so that the slab in Figure 7 is oriented parallel to the coordinate axes. This new orientation has two advantages: multistart can randomly select starting points in a box with much higher density of evaluable points, and numerical differentiation in the directions along the slab will tend to be more accurate since the direction of rapid change normal to the slab will not be mixed into the calculation. Figure 16 shows the results of a multistart CFSQP in this rotated space: the optimizations reach the same two optima as the earlier CFSQP multistart in Figure 6, but the average number of function evaluations required dropped by 25% compared with Figure 6, yielding a corresponding improvement in speed.

start point	takeoff mass	e_1	e_2	l_e	function evaluations	local optimum
1	306524.7	0.95	1.00	57.1	294	A
2	306524.7	0.95	1.00	57.1	661	A
3	306995.1	0.94	1.04	52.7	258	B
4	306524.7	0.95	1.00	57.1	173	A
5	306524.7	0.95	1.00	57.1	169	A
6	306524.7	0.95	1.00	57.1	490	A
7	306524.7	0.95	1.00	57.1	195	A
8	306524.7	0.95	1.00	57.1	157	A
9	306524.7	0.95	1.00	57.1	161	A
10	306524.7	0.95	1.00	57.1	252	A

Figure 17: Multistart CFSQP with nonlinear problem reformulation

Case 2 in Figure 14 is a more complex sort of problem reformulation. The motivation for this reformulation is to create a more intuitive design space. The original design space of three flap lengths gives very little suggestion of what flap lengths might lead to a good design, and as Figures 8 and 9 indicate, the original space has very bad designs which are very "close" to all the good designs. The nonlinear problem reformulation in case 2 of Figure 14 keeps the external flap (see Figure 1) as part of the nozzle design, but instead of specifying the other two flap lengths, the cross-sectional area of the nozzle at its exit is specified in a nondimensional fashion at two points during the aircraft's mission. The nondimensionalization used is to specify a ratio of the actual exit area to the ideal exit area for a perfect nozzle at the same point in the mission. Thus if these two parameters have the value 1, the nozzle will work perfectly at those two points in the mission. The optimizer still needs to vary these values, since the best overall nozzle design for a mission will typically not be best at any particular point in the mission — it is a compromise. Figure 17 shows the results of a multistart CFSQP in this transformed space: the optimizations reach the same two optima as the earlier CFSQP multistart in Figure 6, but the average number of function evaluations required dropped by 20% compared with Figure 6, yielding a corresponding improvement in speed. Note also that CFSQP typically achieves a slightly better design quality in this transformed space by

start point	takeoff mass	e_0	l_c	l_e	function evaluations	local optimum
1	306524.7	0.93	4.3	57.0	202	A
2	306524.7	0.93	4.3	57.1	188	A
3	306526.9	0.93	4.3	57.0	147	A
4	306524.7	0.93	4.3	57.1	114	A
5	306524.7	0.93	4.3	57.1	255	A
6	306524.7	0.93	4.3	57.1	238	A
7	308257.9	0.78	2.2	53.4	245	***
8	306524.7	0.93	4.3	57.1	217	A
9	306524.8	0.93	4.3	57.0	214	A
10	306524.7	0.93	4.3	57.0	185	A

Figure 18: Multistart CFSQP with Case 3 problem reformulation

more accurately computing the global optimum, perhaps because the problem reformulation makes it possible to compute gradients with less roundoff error.

Case 3 in Figure 14 is a compromise between Case 0 and Case 2 — it retains two of the original nozzle flap lengths and only uses one nondimensionalized exit area. Figure 18 shows optimization results in this space. Here the average number of function evaluations required dropped by 43% compared with Figure 6, yielding a corresponding improvement in speed. Note that the 7th stopping point in this example (marked as ***) is not one of the local optima encountered before. In fact, investigation using SST revealed that this stopping point is not a true local optimum at all: at this point the gradient of the objective function points directly into an unevaluable region of the search space, so CFSQP was unable to achieve further local improvements in takeoff mass.

In order to test the generality of the problem reformulations of Figure 14, we also tested their impact on the performance of genetic algorithms [Goldberg 1989] for finding the optimal design for the NDA exhaust nozzle. The algorithm used combined classical methods with new ideas inspired from the search space structure revealed through the use of SST, and gave comparable results to the multistart CFSQP.

The genetic algorithm (GA) we implemented uses selection, mutation and crossover op-

erators to search for the global optimum. Each run of the GA was more expensive than a single CFSQP run, but the higher cost can be justified by the higher degree of confidence in reaching the global optimum.

An individual in the GA was represented by a sequence of three real numbers representing the flap lengths (or other parameters in the case of reformulation), and the fitness of the individual was based on the takeoff mass of the corresponding aircraft design. A starting population was generated at random to start the algorithm.

Selection by rank, a method in which the population is maintained in sorted order according to fitness and the chance of an individual being selected depends on its order rather than its actual fitness, was used. This strategy was used instead of the classical roulette wheel selection strategy due to the relatively narrow fitness range (takeoff mass spectrum) in this problem. If roulette wheel selection were used, the simple fitness measure, which is inversely proportional to takeoff mass, would lead to near random selection and very slow convergence. On the other hand, using a more complicated fitness measure would make the performance dependent to a large extent on the robustness of the measure used, thus introducing unnecessary complexity.

Mutation was done in two different ways, the classical way, which is to make a small random perturbation in the mutated individual's parameter values, and a new way in which the direction of the small perturbation was selected to maximize the likelihood of fitness improvement, based on information obtained from the structure of all the individuals of the population. This new way of mutation proved to be very useful in improving the accuracy of the final result, making it very close to the exact global optimum.

Crossover was also done in two different ways, the classical way, which is to exchange some of the design parameters between two individuals, and a new way in which each of the two sequences of design parameters was regarded as a point in a vector space and crossover was done by selecting a random point along the line formed by joining these two points and taking it to represent the new born individual. This second way was much more appropriate than the classical way for the original search space, and was still a very powerful tool in the

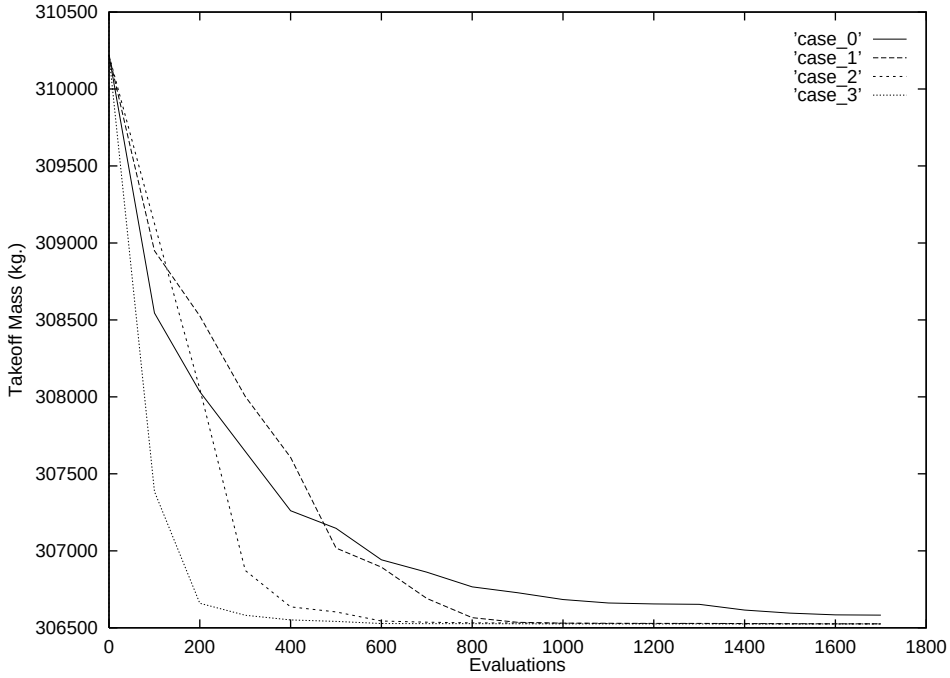


Figure 19: Impact of problem reformulations on design using genetic algorithms

transformed spaces as well.

Figure 19 shows the impact of our problem reformulations on exhaust nozzle design using genetic algorithms. The figure illustrates the takeoff mass of the best design found after a specific number of iterations, averaged over 20 runs of the GA in each space. Note that the GA search performs considerably better in the reformulated spaces than in the original search space. In particular, Figure 19 indicates that GA performance with the case 3 reformulation of Figure 14 is quite impressive, and for finding the global optimum appears competitive with or perhaps superior to CFSQP, which in this multimodal space must be started from several points to ensure a high probability of finding the global optimum. Part of the reason for the improvement of the GA in the reformulated spaces over the original space is that the classical crossover operator was able to make a much larger contribution in these spaces.

6 Conclusion

The Search Space Toolkit (SST) is a suite of tools for investigating the properties of continuous search spaces. SST identifies properties such as number of local optima, local properties in piecewise smooth regions, topology/geometry of the evaluable regions of the objective function, ridges, valleys, and noise. These properties are key characteristics of search spaces which arise in designing complex engineering artifacts whose evaluation requires significant computation by a numerical simulator, and the properties SST investigates can have a major impact on the searchability of design spaces. The search spaces which SST explores differ significantly from the discrete search spaces that typically arise in artificial intelligence research, and properly searching such spaces is a fundamental AI research area. We have extended existing research on unified AI/optimization approaches, focusing on the combination of optimization with ideas from two subfields of AI: search and problem formation.

7 Acknowledgments

This research depended critically on our collaboration with Ron Luffy and Steve Scavo of General Electric Aircraft Engines and Gene Bouchard of Lockheed. We thank Gerard Richter and our other colleagues in the AI/Design research group for valuable contributions to the work described in this paper. This research is 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

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

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.

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

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.

Eugene Charniak and Drew McDermott. *Introduction to Artificial Intelligence.* Addison-Wesley, Reading, Mass., 1987. Reprinted with corrections January, 1987.

T. Ellman and M. Schwabacher. Abstraction and decomposition in hillclimbing design optimization. Technical Report CAP-TR-14, Department of Computer Science, Rutgers University, January 1993.

Andrew Gelsey and Don Smith. A computational environment for exhaust nozzle design. In *Proceedings, Computing in Aerospace 10*, pages 531–539, San Antonio, TX, March 1995. AIAA-95-1016-CP.

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

David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, Mass., 1989.

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.

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.

Jae Kyu Lee and Min Yong Kim. Case-based learning for knowledge-based optimization modeling system: UNIK-CASE. *Expert Systems With Applications*, 6:87–95, 1993.

Jae Kyu Lee and Min Yong Kim. Knowledge-assisted optimization model formulation: UNIK-OPT. *Decision Support Systems*, 13:111–132, 1995.

Jae Kyu Lee and Yong Uk Song. UNIK-PMA: A unifier of optimization with rule-based systems by the post-model analysis. *International Journal of Information Systems in Accounting, Finance, and Management*, 1995. Forthcoming.

Jack D. Mattingly, William H. Heiser, and Daniel H. Daley. *Aircraft Engine Design.* AIAA education series. American Institute of Aeronautics and Astronautics, New York, N.Y., 1987.

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

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

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

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.

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C : the Art of Scientific Computing.* Cambridge University Press, Cambridge [England] ; New York, 2nd edition, 1992.

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.

M. Schwabacher, T. Ellman, and H. Hirsh. Learning when reformulation is appropriate for iterative design. To appear at IJCAI-95 Workshop of Machine Learning in Engineering, 1995.

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

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.

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

Sholom M. Weiss and Casimir A. Kulikowski. *Computer Systems That Learn*. Morgan Kaufmann, San Mateo, CA, 1991.

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.