NASA-CR-204606

INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING, VOL. 39, 1761–1774 (1996)

COMPARATIVE EVALUATION OF DIFFERENT OPTIMIZATION ALGORITHMS FOR STRUCTURAL DESIGN APPLICATIONS

SURYA N. PATNAIK

Ohio Aerospace Institute Cleveland, OH 44142, U.S.A.

RULA M. CORONEOS, JAMES D. GUPTILL AND DALE A. HOPKINS

NASA Lewis Research Center, Cleveland, OH 44135, U.S.A.

SUMMARY

Non-linear programming algorithms play an important role in structural design optimization. Fortunately, several algorithms with computer codes are available. At NASA Lewis Research Centre, a project was initiated to assess the performance of eight different optimizers through the development of a computer code CometBoards. This paper summarizes the conclusions of that research. CometBoards was employed to solve sets of small, medium and large structural problems, using the eight different optimizers on a Cray-YMP8E/8128 computer. The reliability and efficiency of the optimizers were determined from the performance of these problems. For small problems, the performance of most of the optimizers could be considered adequate. For large problems, however, three optimizers (two sequential quadratic programming routines, DNCONG of IMSL and SQP of IDESIGN, along with Sequential Unconstrained Minimizations Technique SUMT) outperformed others. At optimum, most optimizers captured an identical number of active displacement and frequency constraints but the number of active stress constraints differed among the optimizers. This discrepancy can be attributed to singularity conditions in the optimization and the alleviation of this discrepancy can improve the efficiency of optimizers.

KEY WORDS: optimization; algorithms; structural; design; comparative; evaluation

INTRODUCTION

Non-linear programming algorithms play an important role in structural design optimization. Fortunately, several algorithms with computer codes have been developed during the past few decades. To assess the performance of different optimizers, a project was initiated at NASA Lewis Research Centre and a computer code called CometBoards, which is an acronym for Comparative Evaluation Test Bed of Optimization and Analysis Routines for the Design of Structures[1] has been developed. CometBoards incorporates eight popular optimization codes: the Sequential Unconstrained Minimizations Technique (SUMT);[2] the Sequential Linear Programming method (SLP);[3] the Feasible Directions method (FD);[3] the Sequential Quadratic Programming technique (SQP of IDESIGN);[4] the DNCONG of the IMSL routine;[5] NPSOL, which is available in the NAG library;[6] the Reduced Gradient method (RG);[7] and the Optimality Criteria methods (OC).[8] CometBoards was employed to solve a set of 41 structural problems by using its optimizers on a Cray-YMP8E/8128 computer. The reliability and efficiency of the eight optimizers were ascertained on the basis of the performance of these problems. The problems were solved for

CCC 0029–5981/96/101761–14
© 1996 by John Wiley & Sons, Ltd.

Received 13 May 1995
Revised 29 August 1995

multiple load conditions, and behaviour constraints were imposed on stresses, displacements and frequencies. The examples were selected so that at optimum, numerous stress, displacement and frequency constraints were active. Initial design, upper and lower bounds and convergence parameters were specified to ensure that the evaluation had no bias towards any particular optimizer or any particular problem. The eight optimizers might have been updated during the time CometBoards was developed, but any such improvements were not accounted for.

Evaluations of optimizers that are available in the literature[9-18] deal broadly with individual code validation by their developers. The studies lack uniformity because problems and computational platforms differ and the evaluations are over a decade old. For example, Arora, the developer of SQP of IDESIGN, compared his algorithm to the NAG/NPSOL optimizer.[9-11] Most of Arora's problems were trusses for stress and displacement constraints and were solved on a PRIME 750 computer. Schittkowski, who is the developer of the DNCONG optimization routine in the IMSL library, essentially validated his code[12-14] by solving many theoretical examples. Venkayya, one of the developers of ASTROS in which OC and FD optimizers are used,[16] attempted an evaluation of a few practical problems on a VAX 11/785 computer.[15] An intermediate complexity wing problem, used by Venkayya with stress and displacement constraints,[15] is also included in our test bed with the addition of frequency constraint. Ragsdell's evaluation,[17,18] includes mostly simple mechanical application problems. The current paper differs from those available in the literature in several respects: (1) a single tool, CometBoards, evaluates all eight optimizers on a common Cray-YMP computer; (2) solutions to a set of problems, which were grouped into categories of small, medium and large, are used; and (3) design parameters were selected to ensure that the evaluation had no bias towards problems or optimizers. In brief, the comprehensive evaluation presented in this paper does not duplicate previous work. This paper presents a brief theory of optimization methods, a description of CometBoards, a summary of the numerical examples and their solutions, discussion and conclusions.

THEORY OF OPTIMIZATION METHODS

Structural design can be formulated as: Find the n design variables $\vec{x}$ within the prescribed upper and lower bounds ($x_i^L \leqslant x_i \leqslant x_i^U$, $i = 1, 2, \ldots, n$) which make a scalar objective function $f(\vec{x})$ an extremum (here, minimum weight) subject to: a set of m_i inequality constraints $g_j(\vec{x}) \geqslant 0$, ($j = 1, 2, \ldots, m_i$) and m_e equality constraints $g_{j+m_i}(\vec{x}) = 0$ ($j = 1, 2, \ldots, m_e$).

Stress, displacement and frequency behaviour constraints were considered in this study. A cursory account of the different optimization methods available in CometBoards is provided herein. Readers may refer to specified references for details.

SUMT, as implemented in the code NEWSUMT, is available in CometBoards. In NEWSUMT, the penalty function has been modified to improve efficiency and a modified Newton's approach is used to calculate the direction vector while a golden section technique is used to determine step length. SLP, as implemented in the Design Optimization Tools (DOT 2.0), is available in CometBoards. From the original non-linear problem, a linear programming subproblem is obtained by linearizing a set of critical constraints and the objective function around a design point. The linearization process and linear solution sequence is repeated until convergence is achieved. FD, as implemented in DOT 2.0, is available in CometBoards. In FD, a direction is obtained that is both usable and feasible. A minimum along the search direction is generated by polynomial approximation. Three implementations of the sequential quadratic programming technique, SQP of IDESIGN, DNCONG of IMSL and NPSOL in NAG, are available in CometBoards. In this technique, the original non-linear problem is solved through a sequence of quadratic subproblems. In SQP of IDESIGN, a Lagrangian function is

approximated. The step length is obtained by minimizing a composite descent function. DNCONG of IMSL uses quasi-Newton updates for the Hessian of the Lagrangian function while the constraints are linearized.[12, 14, 19] The step length for an augmented Lagrangian is calculated using a bisection method.[20] NPSOL in NAG also uses an augmented Lagrangian. The search direction is generated through a quadratic subproblem while step length is calculated using an augmented Lagrangian, which is designed to avoid discontinuities as much as possible. RG, as implemented in the code OPT,[7, 21–23] has been incorporated into CometBoards. This method partitions the design variable into decision and slave variables and a reduced gradient is used to generate a search direction. A line search is carried out by bounding the minimum and then calculating the minimum within some tolerance.[24–26] OC, available in CometBoards, can be considered as a variant of the Lagrange multiplier approach applied to structural design problems. In OC, an iterative scheme is followed to update the multipliers and the design variables separately.

In addition to the eight popular optimizers (SUMT, SLP, FD, SQP in IMSL, SQP of IDESIGN, NPSOL in NAG, RG and OC) considered in this paper, there are other routines which can be used for design applications. These other routines include, mFD (modified FD),[27] GRG2 (Generalized Reduced Gradient method),[28] CONLIN,[29] MOM (Method Of Multipliers),[30] Genetic,[31] Convex,[32] Asymptotes,[33] dual methods,[34] etc. Mention has been made regarding the potential of genetic and asymptote algorithms. However, to date, the merits for both of these algorithms have been shown for only rather modest problems. For example, the potential of the asymptote algorithm[33] is shown through solutions for a determinate cantilevered beam with a single displacement constraint and for two simple trusses with only a few bars for stress limitations. In brief, the maturity and applicability of some of these algorithms for complex structural systems, under static and dynamic constraints, still remain to be proven, and at this time these algorithms have not been incorporated into the CometBoards test bed.

DESCRIPTION OF COMETBOARDS

The basic organization of CometBoards is depicted in Figure 1. The central executive with command level interface (Figure 1) links the three modules (optimizer, analyser and data input)

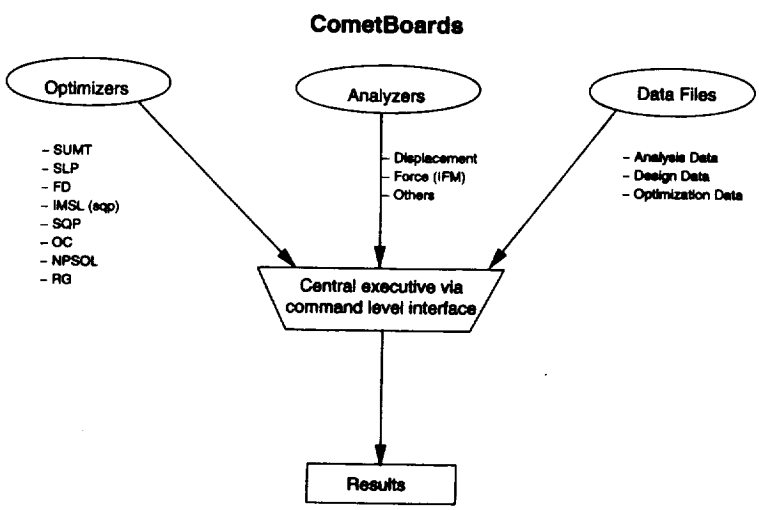

Figure 1. Comparative evaluation test bed of optimization and analysis routines for the design of structures (CometBoards)

Table I. Summary for 41 testbed problems

Problem No.	Problem description and no. of design variables	Constraints specified	SUMT	Active constraints				NPSOL	RG	OC
				SLP	FD	SQP	IMSL			
P1a	3-bar truss (3 IDV, ID = 1)	3S, 2D, 1F	1S, 1D, 1F	1S, 1D, 1F,	1S, 1D, 1F	1S, 1D, IF	1S, 1D, 1F	1S, 1D, 1F	1S, 1D, 1F	*1F
P1b	3-bar truss (3 IDV, ID = OPT)	3S, 2D, 1F	1S, 1D, 1F	1S, 1D, 1F,	1S, 1D, 1F	1S, 1D, IF	1S, 1D, 1F	1S, 1D, 1F	1S, 1D, 1F	*1F
P1c	3-bar truss (3 IDV, ID = 1·5* OPT)	3S, 2D, 1F	1S, 1D, 1F	1S, 1D, 1F,	1S, 1D, 1F	1S, 1D, IF	1S, 1D, 1F	1S, 1D, 1F	1S, 1D, 1F	*1F
P1d	3-bar truss (3 IDV, ID = 0·5* OPT)	3S, 2D, 1F	1S, 1D, 1F	1S, 1D, 1F,	1S, 1D, 1F	1S, 1D, IF	1S, 1D, 1F	1S, 1D, 1F	1S, 1D, 1F	*1F
P2	Tapered 10-bar truss (10 IDV)	20S, 4D, 1F	7S, 2D, 1F	*—	*—	8S, 2D, 1F	8S, 2D, 1F	8S, 2D, 1F	*1S, 1F	*5S, 2D, 1F
P3	Tapered contilever beam (8 IDV)	16S,4D, 1F	6S	6S	6S	6S	6S	6S	6S	4S
P4	25-bar truss (8 LDV)	50S, 36D	4D	4D	4D	4D	4D	4D	*2D	*—
P5	165-ft-tall antenna tower (6LDV)	504S, 24D, 1F	*7S, 12D, 1F	6S, 12D, 1F	6S, 12D, 1F	6S, 12D, 1F	6S, 12D, 1F	6S, 12D, 1F	*1F	*—
P6	60-bar trussed ring (25 LDV)	180S, 3D, 1F	21S, 1D, 1F	22S, 1D, 1F	20S, 1D, 1F	21S, 1D, 1F	21S, 1D, 1F	19S, 1D, 1F	*18S, 1D, 1F	*10S, 1F
P7	Geodesic dome (12 LDV)	252S, 1D, 1F	162S, 1D, 1F	*148S, 1D, 1F	*156S, 1D, 1F	168S, 1D, 1F	156S, 1D, 1F	162S, 1D, 1F,	*18S, 1D	*12S
P8a	Intermediate complexity wing (57 LDV, ID = 1·0)	316S, 4D, 1F	106S, 1D	*47S	93S, 1D	117S, 1D, 1F	117S, 1D, 1F	75S, 1D, 1F	*6S	*19S
P8b	Intermediate complexity wing (57 LDV, 1D = OPT)	316S, 4D, 1F	*—	106S, 1D	106S, 1D	106S, 1D	117S, 1D, 1F	117S, 1D, 1F	106S, 10	*42S
P8c	Intermediate complexity wing (57 LDV, ID = 1·5*OPT)	316S, 4D, 1F	109S, 1D	*56S, 1F	*7S	*115S, 1D, 1F	*88S, 1D, 1F	*2S	*7S	*—
P8d	Intermediate complexity wing (57 LDV, ID = 0·5* OPT	316S, 4D, 1F	117S, 1D, 1F	*13S	*13S	*99S, 1D, 1F	*—	*14S	*7S	*—
P9	10-bar truss (10 IDV)	20S, 4D, 1F	7S, 2D, 1F	*—	*—	8S, 2D, 1F	8S, 2D, 1F	8S, 2D, 1F	*1F	*5S, 2D, 1F
P10	10-bar truss (5 LDV)	20S, 4D, 1F	6S, 2D, 1F	*—	*—	6S, 2D, 1F	6S, 2D, 1F	6S, 2D, 1F	*3S, 1F	*6S, 2D, 1F
P11	Stiffened 10-bar truss (18 IDV)	36S, 4D, 1F	8S	9S	9S	9S	8S	9S	*1S	5S
P12	Stiffened 10-bar truss (2 LDV)	36S, 4D, 1F	2S	2S	2S	2S	2S	2S	*1F	2S
P13	Cantilever membrane (8 IDV)	16S, 4D, 1F	6S	6S	6S	6S	6S	6S	6S	4S
P14	Cantilever membrane (1 LDV)	16S, 4D, 1F	1S	1S	1S	1S	1S	1S	1S	1S
P15	Cantilever membrane 16-quad elements (16 IDV)	32S, 4D, 1F	1S, 4D	4D	1S, 3D	*—	4D	*4S	*5S	4D
P16	Cantilever membrane 32-quad elements (16 LDV)	64S, 4D, 1F	1S, 4D	*—	*—	4D	4D	*9S	*4S	4D
P17	Cantilever membrane 48-quad elements (16 LDV)	96S, 4D, 1F	13S, 4D	*—	*—	*13S, 4D	15S, 4D	*8S	*3S	13S, 2D
P18	Cantilever membrane 64-quad elements (16 LDV)	128S, 4D, 1F	32S, 4D	*5D	*—	*29S, 4D	31S, 4D	*16S	*—	23S, 2D

P19	60-bar trussed ring (25 LDV)	180S	38S	*11S	*33S	*35S	38S	38S	*14S	40S
P20	60-bar trussed ring (25 LDV)	3D	1D	1D	1D	1D	1D	1D	*1D	1D
P21	60-bar trussed ring (25 LDV)	1F	*—	*1F	1F	1F	1F	1F	*1F	*1F
P22	60-bar trussed ring (25 LDV)	180S, 24D	28S, 1D	26S, 1D	27S, 1D	*30S, 1D	27S, 1D	29S, 1D	*20S	18S, 1D
P23	60-bar trussed ring (25 LDV)	24D, 1F	*1D, 1F	*1D, 1F	1D, 1F	1D, 1F	1D, 1F	*1D, 1F	*1F	*1D, 1F
P24	Stiffened 60-bar trussed ring (49 LDV)	252S	75S	*27S	*62S	75S	75S	75S	75S	*59S
P25	Stiffened 60-bar trussed ring (49 LDV)	3D	1D	*1D	1D	*1D	1D	*—	1D	1D
P26	Stiffened 60-bar trussed ring (49 LDV)	1F	1F	*1F	*1F	1F	1F	*—	1F	1F
P27	Stiffened 60-bar trussed ring (49 LDV)	252S, 24D	75S, 1D	*27S	*62S, 1D	75S, 1D	75S, 1D	75S, 1D	76S, 1D	*17S
P28	Stiffened 60-bar trussed ring (49 LDV)	24D, 1F	1D, 1F	*1D, 1F	*1D, 1F	1D, 1F	1D, 1F	*1D	1D, 1F	1D, 1F
P29	Stiffened 60-bar trussed ring (49 LDV)	252S, 3D, 1F	44S, 1F	*19S, 1F	*31S, 1F	46S, 1F	46S, 1F	46S, 1F	47S, 1F	*3S
P30	Stiffened ring (24 IDV)	72S	28S	25S	23S	28S	28S	28S	27S	28S
P31	Stiffened ring (24 IDV)	3D	1D	1D	1D	1D	1D	*—	1D	1D
P32	Stiffened ring (24 IDV)	1F	1F	1F	1F	1F	1F	*—	*1F	1F
P33	Stiffened ring (24 IDV)	72S, 24D	28S, 1D	25S, 1D	23S, 1D	27S, 1D	28S, 1D	28S, 1D	25S, 1D	*185, 1D
P34	Stiffened ring (24 IDV)	24D, 1F	1D, 1F	1D, 1F	1D, 1F	1D, 1F	1D, 1F	*2D	1D, 1F	1D, 1F
P35	Stiffened ring (24 IDV)	72S, 3D, 1F	17S, 1F	17S, 1F	17S, 1F	17S, 1F	17S, 1F	17S, 1F,	17S, 1F	*165, 1F

IDV: independent design variables; ID: initial design; LDV: linked design variables; OPT: SUMT optimum design; S: stress constraints; D: displacement constraints; F: frequency constraints

* Optimum weight obtained differs by more than 5 per cent or constraint violation more than 1 per cent (see Reference 1)

of the code to formulate and solve an optimization problem. Note that there are eight choices for optimizers. The analyser options are the displacement method,[8, 35] the integrated force method, [8,36] the simplified force method,[8] etc. There are three input data files, one for analysis (anldat), one for design (dsgndat) and one for optimization (optdat). CometBoards has considerable flexibility in solving a design problem by choosing any one of the eight optimizers and any one of the three analysers. A more detailed description of CometBoards can be found in Reference 1.

EXAMPLE PROBLEMS

The numerical testbed of CometBoards includes over 41 problems, most of which were taken from the literature.[1, 8, 15, 37–42] Minimum weight was the objective and a linking strategy was followed to reduce the number of design variables. Stress, displacement and frequency behaviour constraints were considered. Multiple static load conditions and consistent elemental mass for dynamic analyses were also considered. The load conditions, mass distributions, and behaviour limitations were selected to ensure that several types of behaviour constraints were active at the optimum. The initial design of unity was considered for all problems unless otherwise specified. A consistent set of upper and lower bounds was specified for each problem. Typically, default optimization parameters and convergence criteria specified in the individual codes were used. These parameters, however, were changed when convergence difficulty was encountered. Results for all 41 examples are summarized in Table I. The normalized optimum weight and the normalized Cray-YMP8E/8128 CPU time for a select set of 14 examples are given in Table II and depicted in Plates 1 through 4. The weight was normalized with respect to the optimum weight obtained for the best feasible design. A brief description of the 14 examples follows.

Examples (P1a–P1d). Three-bar truss

The popular three-bar truss[8, 37, 41] (with modulus $E = 30\,000$ ksi and density $\rho = 0{\cdot}1$ lb/in^3.) was subjected to a single load condition. It had three design variables and six constraints (three stress, two displacement and one frequency). Optimum weight and CPU time are depicted in Table II (P1a, P1b, P1c, P1d) and Plates 1 and 4. The optimum weight was 92·87 lb and three constraints (one stress, one displacement and one frequency) were active. Seven optimizers (SUMT, SLP, FD, SQP, IMSL, NPSOL and RG) performed satisfactorily. OC was inadequate, yielding a 38·6 per cent overdesign. The problem was solved again for three different initial designs (the SUMT optimum design, 150 per cent of SUMT optimum and 50 per cent of SUMT optimum). Results followed the earlier pattern where the initial design was unity. The CPU times on the Cray-YMP computer required for different optimizers are depicted in Plate 4. For unit initial design, SLP required the least CPU time of 0·07 s, while RG was most expensive at 3·18 s.

Example P2. Tapered 10-bar truss

A tapered 10-bar aluminum truss[8] was subjected to two load conditions. It had 10 design variables and 25 behaviour constraints (20 stress, four displacement and one frequency). The optimum weight was 3326·74 lb with 11 active constraints (eight stress, two displacement, and one frequency). Four optimizers (SUMT, SQP, IMSL and NPSOL) converged for this example. SLP and FD converged to an 82·4 per cent under-design condition. Optimizer RG failed and OC was marginal at a 5·6 per cent over-design. Cray-YMP CPU time varied between 1·28 s for NPSOL and 1·91 s for SUMT.

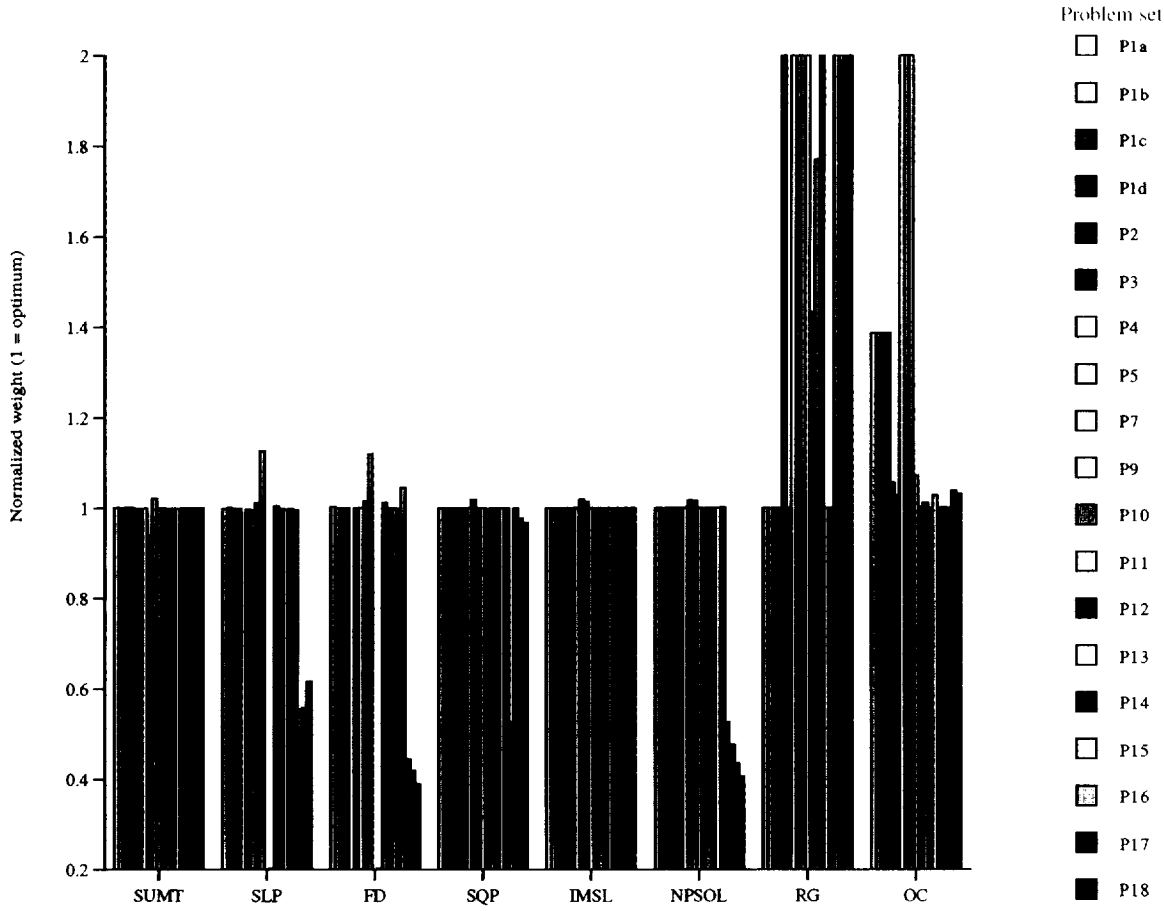

Plate 1 Performance of different optimizers for small problems

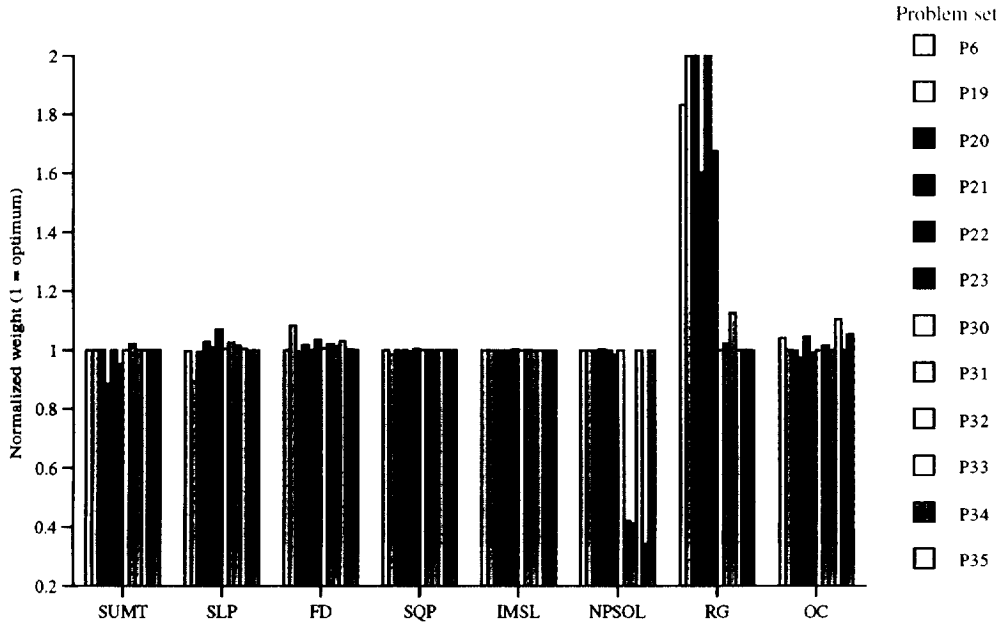

Plate 2 Performance of different optimizers for medium problems

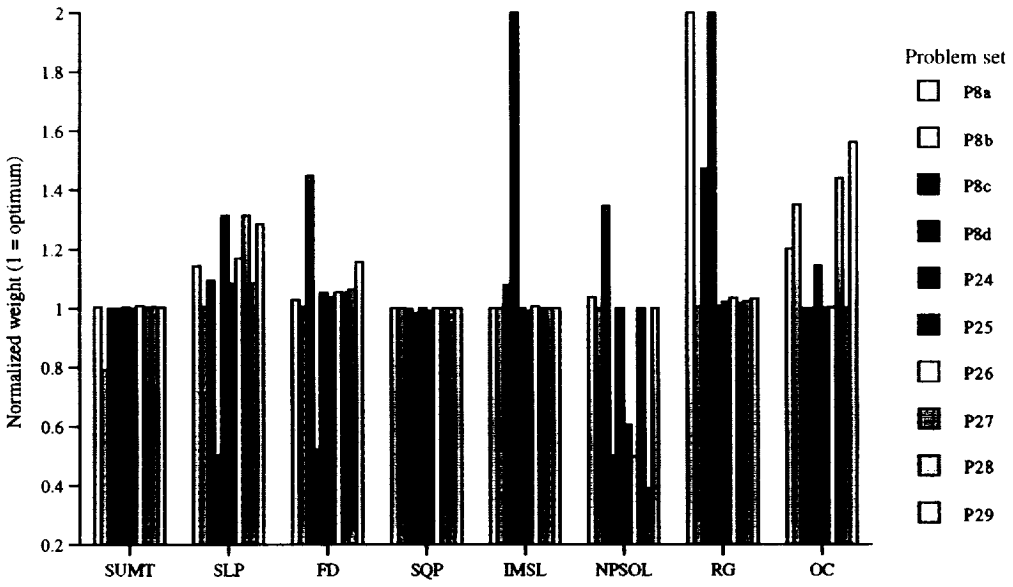

Plate 3 Performance of different optimizers for large problems

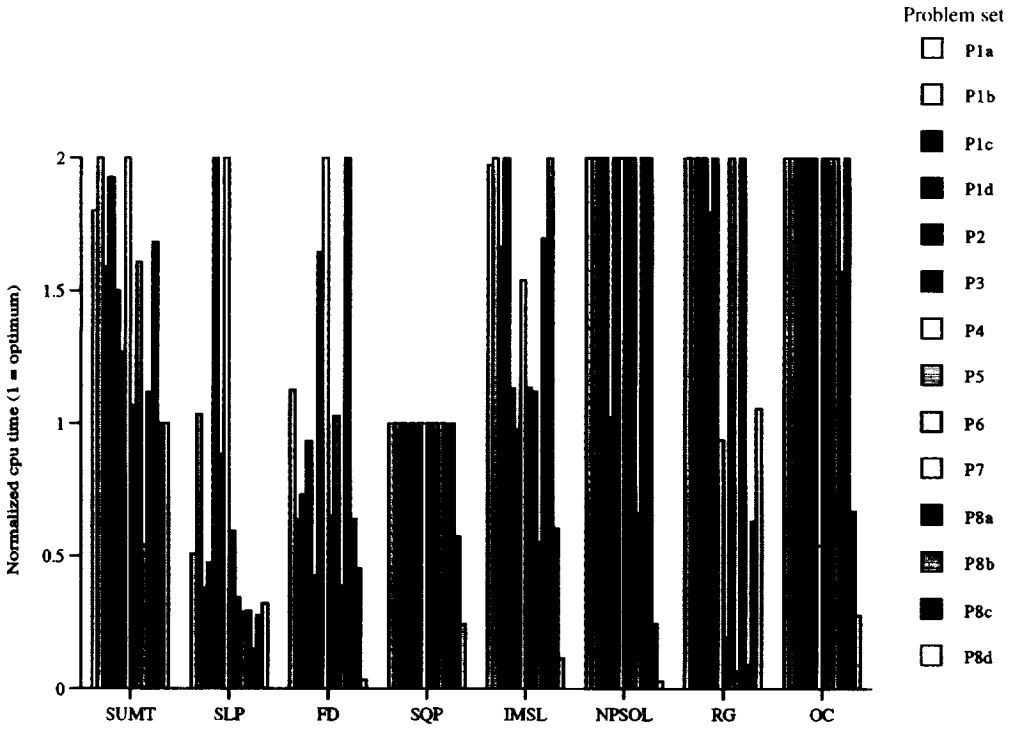

Plate 4 Cray-YMP CPU time for different optimization methods for 14 problems

Table II. Optimum weight and Cray-YMP 8E/8128 CPU time for a selected set of example problems

	Optimization methods															
Problem number	SUMT Weight**	SUMT CPU	SLP Weight**	SLP CPU	FD Weight**	FD CPU	SQP Weight**	SQP CPU	IMSL Weight**	IMSL CPU	NPSOL Weight**	NPSOL CPU	RG Weight**	RG CPU	OC Weight**	OC CPU
P1a	1·001	1·799	0·999	0·507	1·003	1·125	1·000	1·000	1·000	1·972	1·000	2·076	1·000	22·069	1·386	10·257
P1b	1·001	7·833	1·001	1·033	1·001	0·633	1·000	1·000	1·000	9·633	1·000	155·267	1·000	5·567	1·386	49.233
P1c	1·001	1·588	0·999	0·378	0·999	0·730	1·000	1·000	1·000	1·662	1·000	31·818	1·000	2·000	1·386	10·000
P1d	1·001	1·926	0·999	0·474	1·000	0·933	1·000	1·000	1·000	2·733	1·000	14·444	1·000	16·452	1·386	10·985
P2	1·000	1·500	*0·176	3·396	*0·176	0·424	1·000	1·000	1·000	1·133	1·000	1·022	(Failed)	—	1.056	9.324
P3	0·999	1·268	0·977	0·883	1·000	1·643	1·000	1·000	1·000	0·976	1·001	3·169	1·000	6·121	1·028	16·834
P4	1·000	2·533	0·996	3·596	1·001	2·184	1·000	1·000	1·000	1·539	1·000	3·759	(Failed)	—	(Failed)	—
P5	*0·940	1·065	1·012	0·591	1·015	0·646	1·019	1·000	1·019	1·134	1·017	5·026	(Failed)	—	2·773	4·62
P6	1·000	1·605	0·997	0·343	0·999	1·026	1·000	1·000	1·000	1·120	1·000	3·899	1·832	20·716	*1·041	8·279
P7	1·021	0·538	1·127	0·287	1·120	0·385	1·000	1·000	1·015	0·550	1·016	0·658	2·022	0·066	2·976	4·456
P8a	1·004	1·116	1·143	0·292	1·029	2·910	1·000	1·000	1·000	1·695	1·037	7·712	(Failed)	—	1·201	1·571
P8b	0·790	1·680	1·004	0·148	1·004	0·635	1·000	1·000	1·000	5·884	1·000	3·323	1·004	0·089	1·350	8·076
P8c	1·000	1·000	1·095	0·276	1·448	0·452	*0·998	0·571	*1·077	0·602	1·346	0·244	1·471	0·627	1·000	0·666
P8d	1·000	1·000	*0·502	0·321	*0·521	0·033	*0·981	0·244	(Failed)	—	*0·501	0·028	(Failed)	—	1·000	0·276

*Infeasible design
**Normalized weight

Example P3. Tapered cantilever beam

The cantilever truss of Example P2 was modelled next using eight triangular membrane elements.[43] The loads and constraints were kept identical to those in Example P2. There are eight thicknesses of the elements which were considered the eight design variables. The problem had 21 constraints (16 von Mises stress, four displacement and one frequency). Optimum results obtained are given in Table II. The optimum weight was 1440·24 lb with six active stress constraints. Seven optimizers (SUMT, SLP, FD, SQP, IMSL, NPSOL and RG) performed well while OC produced a 2·8 per cent over-design (see Table II). Cray-YMP CPU time varied from 1·62 s for SLP to 11·22 s for RG.

Example P4. 25-bar truss

A 25-bar aluminum truss[37, 38] had eight linked design variables and was subjected to two load conditions. It had a total of 86 behaviour constraints (50 stress and 36 displacement). Six optimizers (SUMT, SLP, FD, SQP, IMSL and NPSOL) converged to an optimum weight of 544·73 lb with four active displacement constraints (Tables I and II). Optimizers RG and OC failed. Cray-YMP CPU time ranged from 1·64 s for SQP to 6·15 s for NPSOL.

Example P5. 165-ft-tall antenna tower

A 165-ft-tall steel antenna tower with 252 members[37] had six linked design variables and was subjected to two load conditions. Its overhead dish antenna was modelled as a lumped mass for frequency calculations. It had a total of 529 behaviour constraints (504 stress, 24 displacement and one frequency). Five optimizers (SLP, FD, SQP, IMSL and NPSOL) converged to an optimum solution of 5299·84 lb with small deviations (Table II). At the optimum, six stress, 12 displacement and one frequency constraints are active. Optimizers RG and OC failed while SUMT produced a 6 per cent under-design. The Cray-YMP CPU time varied between 222·71 s for SLP and 1893·80 s for NPSOL.

Example P6. 60-bar trussed ring

A 60-bar trussed aluminum ring[8] was subjected to three load conditions and had two lumped masses. It had a total of 184 constraints (180 stress, three displacement and one frequency) and 25 linked design variables. The optimum weight was 414.51 lbs, and at optimum, 22 stress, one displacement and one frequency constraints were active. Six optimizers (SUMT, SLP, FD, SQP, IMSL and NPSOL) converged (Table II). Optimizer RG failed, whereas OC produced a 4·1 per cent over-design with a 1·1 per cent constraint violation. Cray-YMP CPU solution time ranged from 12·67 s for SLP to 144·11 s for NPSOL.

Example P7. Geodesic dome

A geodesic dome,[39, 40] with a diameter of 240 in. and a height of 30 in., was subjected to a single load condition. It was modelled using 156 bars and 96 triangular membrane elements. The bars were made of a material with modulus $E = 30\,000$ ksi, and density $\rho = 0{\cdot}1$ lb/in^3. Membranes were made of aluminum, with modulus $E = 10\,000$ ksi, and density $\rho = 0{\cdot}1$ lb/in^3. The bar areas and membrane thicknesses were grouped to obtain eight and four linked design variables, respectively. The dome had a total of 254 constraints (156 stresses for bars, 96 von Mises stresses

for membranes, one displacement and one frequency). The optimum weight obtained was 1022·67 lb with 170 active constraints (168 stress constraints, one displacement and one frequency (see Table I). Four optimizers (SUMT, SQP, IMSL and NPSOL) converged with small deviations. Optimizers RG and OC failed, while SLP and FD produced over-designs (12·7 and 12·0 per cent, respectively). The Cray-YMP CPU time varied between 448·32 s for SUMT and 548·36 s for NPSOL.

Examples (P8a–P8d). Intermediate complexity wing

An intermediate complexity wing[8, 15] was modelled with a total of 158 elements consisting of 39 bars, two triangular membranes, 62 quadrilateral membranes and 55 shear panels. The wing is made of aluminum with modulus $E = 10\,500$ ksi and density $\rho = 0{\cdot}1$ lb/in^3. The elements were grouped to obtain 57 linked design variables. The wing, which was subjected to two load conditions, had a total of 321 behaviour constraints (316 stress, four displacement and one frequency). The optimum design for this problem was obtained from four different initial points: (1) initial design of unity; (2) initial design equal to the SUMT optimum design; (3) initial design equal to 150 per cent of the SUMT optimum design; and (4) initial design which is infeasible at 50 per cent lower than the SUMT optimum design. Results obtained for all four cases are summarized in Table II (P8a–P8d). The optimum design was 387·76 lb and there were a total of 119 active constraints (117 stress, one displacement and one frequency). For initial design equal to unity (see Table II, Problem P8a), optimizers SUMT, FD, SQP, IMSL and NPSOL reached the optimum within a 3·7 per cent error margin. Optimizer RG failed to solve the problem. Optimizers SLP and OC also failed to converge to the optimum (producing 14·3 and 20·1 per cent over-designs, respectively). Cray-YMP CPU time varied between 1075·21 s for SQP and 8292·35 s for NPSOL.

DISCUSSION

For the purpose of this discussion, the 41 examples of the CometBoards test bed are grouped as small, medium and large problems. The number of linked design variables ranged between 3 and 19 for small problems (Group I). Group I contains a total of 19 problems, which are designated as P1a–P5, P7 and P9–P18. The normalized optimum weight for small problems obtained by each optimizer is depicted in Plate 1. For medium problems (Group II), the number of linked design variables ranged between 20 and 39. There are 12 medium problems, which are designated as P6, P19–P23 and P30–P35. The normalized optimum weight for the medium problems obtained by each optimizer is illustrated in Plate 2. Problems with more than 40 independent design variables are referred to as large problems (Group III). There are 10 large problems which are designated as P8a–P8d, and P24–P29. The normalized optimum weight for large problems obtained by each optimizer is depicted in Plate 3.

The discussion is separated into the following five categories: (1) convergence to the optimum weight, (2) number of active constraints at optimum, (3) Cray-YMP8E/8128 CPU time required to solve the problem, (4) singularity in structural optimization and (5) default optimization parameters.

Convergence to optimum weight

The normalized optimum weights for all 41 problems, obtained by the eight optimizers, are depicted in Plates 1–3 for small, medium and large problems, respectively. In these Plates, unity

represents optimum weight and more than unity indicates over-design, while less than unity is infeasible design. For the purpose of comparison, a solution with constraint violation of less than 1 per cent and weight which is within 1 per cent of the best feasible design is considered correct. A design is acceptable when the constraint violation is less than 1 per cent and the weight is within 5·0 per cent of the best obtained by the eight optimizers. Convergence to optimum weight for each of the eight optimizers follows.

(1) SUMT converged to optimum solution for 35 of 41 examples, which consisted of 17 small, nine medium, and nine large problems. SUMT failed for four problems. These are: one small problem (P5), two medium problems (P21 and P23) and one large problem (P8b). For the two medium problems, the SUMT solution was more than 1 per cent infeasible. For the large problem, SUMT gave an over-design of more than 5 per cent.
(2) SLP of DOT 2.0 successfully solved 19 of 41 examples, which consisted of 11 small, seven medium, and one large problems. SLP failed for seven small problems (P2, P7, P9, P10 and P16–P18), three medium problems (P19, P21 and P23) and nine large problems (P8a, P8c, P8d and P24–P29).
(3) FD of DOT 2.0, successfully solved 16 of 41 examples, which consisted of nine small, six medium and one large problems. FD produced infeasible designs for six small problems (P2, P9, P10 and P16–P18) and one large problem (P8d). It produced over-design conditions for eight problems (P7, P8c, P19, P24 and P26–P29).
(4) SQP of IDESIGN successfully solved 32 of 41 examples, which consisted of 15 small, 10 medium and seven large problems. This optimizer failed to give a feasible optimum design for three small problems (P15, P17 and P18), two medium problems (P19 and P22) and three large problems (P8c, P8d and P25).
(5) IMSL optimizer DNCONG successfully solved 37 of 41 examples, which consisted of 17 small, 12 medium and eight large problems. DNCONG of IMSL failed to optimize the intermediate complexity wing (Problems P8c and P8d).
(6) NPSOL successfully solved 25 of 41 examples, which consisted of 13 small, eight medium and four large problems. This optimizer failed (with a 1 per cent infeasible constraint) for four small problems (P15–P18), four medium problems (P23, P31, P32 and P34) and four large problems (P8d, P25, P26 and P28). It produced more than 5 per cent over-design for large problem P8c.
(7) RG successfully solved 13 of 41 examples, which consisted of seven small, four medium and two large problems. RG failed for 12 small problems. It also failed for seven medium problems and three large problems. The optimizer RG failed with well over 100 per cent error in the optimum weight for 15 problems.
(8) OC successfully solved 16 of 41 examples, which consisted of six small, five medium and five large problems. OC failed for nine small, two medium and five large problems with an error in the optimum weight exceeding 5 per cent, as well as for three medium problems with an infeasible design greater than 1 per cent.

Number of active constraints at optimum

The number of active constraints at the optimum for all examples is given in Table I. Typically, different optimizers produced identical numbers of active frequency and active displacement constraints. However, the number of active stress constraints generated depended on the optimizer of choice. For example, with the geodesic dome problem (P7), the number of active stress constraints produced were 168 by SQP of IDESIGN, 162 by SUMT and NPSOL, and 156 by

IMSL. Consider next the set of five examples depicted in Table III that failed to converge. The minimum weights ranged between 3·2 and 12·7 per cent over-designs or under-designs. These examples produced correct numbers of displacement and frequency constraints, but failed to produce the correct numbers of active stress constraints. The deficiency in the number of active stress constraints ranged between 3 for Problem P2 and 42 for Problem P8a. For these problems the failure of the optimizers could be attributed to their inability to produce the correct number of active stress constraints. This aspect is also described in the section entitled, 'Singularity in structural optimization' of this paper.

CPU time required for the solution

The normalized CPU times on a Cray-YMP8E/8128 computer were recorded for a set of 14 examples. The normalization was with respect to SQP of IDESIGN except for Problems P8c and P8d, which were normalized with respect to SUMT (Table II and Plate 4). CPU time required to solve the same problem differed among optimizers. Even for a small problem (P1a), normalized CPU time differed from 0·507 for SLP to 22·069 for RG. For a medium problem (P6), normalized time differed between 0·343 for SLP to 3·899 for NPSOL. For a large problem (P8a), normalized CPU time varied from 1·695 for IMSL to 1·116 for SUMT and 1·000 for SQP of IDESIGN. It is observed that variation in CPU time was rather mild for large problems.

Singularity in structural optimization

Singularity in structural optimization was identified for three situations:[8, 41] (1) The number of active constraints exceeds the number of design variables. Out of the 41 test bed problems, the 14 examples listed in Table IV are prone to this type of singularity. (2) Linear functional dependencies exist among a small number of active stress constraints. This type of singularity is suspected to have occurred for some of the examples given in Table III. (3) Linear functional dependencies exist among a small number of active stress and displacement constraints. The identification of this type of singularity by mere inspection may be difficult.

Table III. Five examples that failed to reach optimum weight versus best feasible design

Problem number	Optimization method	Percent over-design	Number of active constraints at optimum versus best feasible design		
			Frequency	Stress	Displacement
P2	OC	5·6	1	5	2
	vs. SQP	0·0	1	8	2
P6	RG	83·2	1	18	1
	vs. SQP	0·0	1	21	1
P7	SLP	12·7	1	148	1
	vs. SQP	0·0	1	168	1
P8a	NPSOL	3·2	1	75	1
	vs. IMSL	0·0	1	117	1
P8c	IMSL	7·7	1	88	1
	vs. SUMT	0·0	1	109	1

Table IV. Problems with active constraints exceeding the number of design variables (singularity can occur in each of these problems)

Problem number, description	Number of design variables	Number of active constraints at optimum
P2, Tapered 10-bar truss	10	11
P5, Antenna tower	6	20
P7, Geodesic dome	12	170
P8a, Intermediate complexity wing	57	119
P9, 10-bar truss	10	11
P10, 10-bar truss	5	9
P17, Cantilever membrane	16	19
P18, Cantilever membrane	16	35
P19, 60-bar trussed ring	25	38
P22, 60-bar trussed ring	25	30
P24, Stiffened ring	49	75
P27, Stiffened 60-bar trussed ring	49	76
P30, Stiffened ring	24	28
P33, Stiffened ring	24	28

It is suggested that code developers should address the singularity issue. Singularity alleviation as discussed in References 8, 41 and 42 can reduce computation and improve reliability of optimizers.

Default optimization parameters

Default parameters (such as convergence criteria, step length, stopping criteria, active constraint region, iteration limitations, etc.) specified by individual optimization codes were used to solve the problems. When a problem failed, the default parameters were changed according to the instructions specified in the user's manual of individual codes in an attempt to successfully solve the problem. In the solution of the 41 test bed problems, it was necessary to change the default optimization parameters quite often in order to reach the correct solution. On an overall basis, default parameters of SUMT, SLP, FD, SQP and IMSL algorithms were adequate for the solution of most problems. Most of the default parameters for RG and NPSOL were changed to improve their performances.

CONCLUSIONS

None of the eight optimizers could successfully solve all the problems. Most optimizers, however, can solve at least one third of the examples. For large problems, the Cray-YMP CPU time was comparable among the optimizers that succeeded. Alleviation of singularity that can occur in structural optimization can improve the optimizer efficiency.

A single winner which can be called most reliable and efficient could not be identified. Overall, three optimizers (IMSL, SUMT and SQP of IDESIGN) scored high marks. For small problems, five optimizers (IMSL, SUMT, SQP of IDESIGN, NPSOL and SLP) satisfactorily solved more than fifty per cent of the problems. For medium problems, six optimizers (IMSL, SQP of IDESIGN, SUMT, NPSOL, SLP and FD) produced correct solutions for at least half of the problems. For large problems, three optimizers (IMSL, SUMT and SQP of IDESIGN) were found to be reliable and efficient.

REFERENCES

1. J. D. Guptill, R. M. Coroneos, S. N. Patnaik, D. Hopkins and L. Berke, *CometBoards User's Manual*, NASA TM-4537, 1996.
2. H. Miura and L. A. Schmit Jr., *NEWSUMT: A FORTRAN Program for Inequality Constrained Function Minimization, User's Guide*, NASA CR-159070, 1979.
3. G. N. Vanderplaats, *DOT User's Manual*, Version 2.04, VMA Engineering, 1989.
4. J. S. Arora, *IDESIGN User's Manual*, Version 3.5.2, Optimal Design Laboratory, University of Iowa, 1989.
5. *IMSL User's Manual, MATH/LIBRARY: FORTRAN Subroutines for Mathematical Applications.*,Vol. 3, Chapter 8, 1987, p. 903.
6. *NAG FORTRAN Library Manual-MARK 15: E04UCF*, NAG Fortran Library Routine Document, Vol. 4, 1991.
7. G. A. Gabriele and K. M. Ragsdell, *OPT-A Nonlinear Programming Code in FORTRAN Implementing the Generalized Reduced Gradient Method, User's Manual*, University of Missouri-Columbia, 1984.
8. S. N. Patnaik, J. D. Guptill and L. Berke, *Merits and Limitations of Optimality Criteria Method for Structural Optimization*, NASA TP-3373, 1993.
9. P. B. Thanedar, J. S. Arora, C. H. Tseng, O. K. Lim and G. J. Park, 'Performance of some SQP algorithms on structural design problems—sequential quadratic programming', *Int. j. numer. methods eng.*, **23**, 2187–2203 (1986).
10. C. H. Tseng and J. S. Arora, 'On implementation of computational algorithms for optimal design 1: preliminary investigation', *Int. j. numer. methods eng.*, **26**, 1365–1382 (1988).
11. C. H. Tseng and J. S. Arora, 'On implementation of computational algorithms for optimal design 2: extensive numerical investigation', *Int. j. numer. methods eng.*, **26**, 1383–1402 (1988).
12. K. A. Schittkowski, 'A numerical comparison of 13 nonlinear programming codes with randomly generated test problems', *Numer. Optim. Dyn. Systems*, 213–234 (1980).
13. W. Hock and K. Schittkowski, 'A comparative performance evaluation of 27 nonlinear programming codes', *Computing*, **30**, 335–358 (1983).
14. K. Schittkowski, C. Zillober and R. Zotemantel, 'Numerical comparison of nonlinear programming algorithms for structural optimization, *Struct. Optim.*, **7**, 1–19 (1994).
15. R. A. Canfield, R. V. Grandhi and V. B. Venkayya, 'Optimum design of structures with multiple constraints', *AIAA J.*, **26**, 78–85 (1988).
16. D. J. Neill, E. H. Johnson and D. L. Herendeen, *Automated Structural Optimization System (ASTROS). AFWAL-TR-88-3028, Wright Patterson Air Force Base vol. II- User's Manual*, 1988.
17. K. M. Ragsdell, 'A survey of some useful optimization methods', in A. H. Soni and H. Atmaran (eds.), *Proc. Design Engineering Technical Conference*, New York, 1974, pp. 129–135.
18. K. M. Ragsdell, 'The evaluation of optimization software for engineering design', *Lecture Notes in Econom. and Math. Systems*, Vol. 199, 1982.
19. B. Pshenichny, 'Algorithms for the general problem of mathematical programming', *Kibernetica*, **5**, 120–125 (1970).
20. K. Schittkowski, 'NLLPQL: A FORTRAN subroutine solving constrained nonlinear programming problems, *Ann. Oper. Res.*, **5**, 485–500 (1986).
21. P. Wolfe, in J. Abadie (ed.), *Methods for Linear Constraints, Nonlinear Programming*, North-Holland, Amsterdam, 1967, pp. 99–131.
22. P. Wolfe, in R. L. Graves and P. Wolfe (ed.), *Methods of Nonlinear Programming, Recent Advances in Math. Programming*, McGraw-Hill, New York, 1963, pp. 76–77.
23. J. Abadie and J. Carpentier, in R. Fletcher (ed.), *Generalization of the Wolfe Reduced Gradient Method to the Case of Nonlinear Constraints, Optimization*, Academic Press, New York, 1969, p. 37.
24. R. Fletcher and C. M. Reeves, 'Function minimization by conjugant gradients', *Comp. J.*, **6**, No. 2, 163-168 (1963).
25. R. L. Fox, *Optimization Methods for Engineering Design*, Addison-Wesley, Reading, MA, 1971.
26. D. M. Himmelblau, *Applied Nonlinear Programming*, McGraw-Hill, New York, 1972.
27. A. D. Belegundu, L. Berke and S. N. Patnaik, 'An optimization algorithm based on the method of feasible directions', *Struct. Optim. J.*, **9**, 83–88 (1995).
28. L. S. Lasdon and A. D. Waren, *GRG2 User's Guide*, University of Texas at Austin, 1986.
29. C. Fleury,'The CONLIN program: a new capability for structural optimization within PERMAS', *Proc. of Finite Elements in Engineering Applications*, Strasbourg, France, 1990.
30. A. Mifle, P. E. Moseley, A. V. Levy and G. M. Coggins, 'On the method of multipliers for mathematical programming problems', *J. Optim. Theory Appl.*, **10**, 1–33 (1972).
31. D. E. Goldberg, *Genetic Algorithms in Search Optimization, and Machine Learning*, Addison-Wesley, Reading, MA, 1989.
32. C. Fleury, 'Structural weight optimization by dual methods of convex programming', *Int. j. numer. methods eng.*, **14**, 1761–1783 (1979).
33. K. Svanberg, 'The method of moving asymptotes—A new method for structural optimization' *Int. j. numer. methods. eng.*, **24**, 359–373 (1987).
34. C. Fleury and V. Braibant, 'Structural optimization: a new dual method using mixed variables', *Int. j. numer. methods. eng.*, **23**, 409–428 (1986).
35. V. B. Venkayya and V. A. Tischler, 'ANALYZE: analysis of aerospace structures with membrane elements', *Tech. Rep. AFFDL-TR-78-170* (1978).

36. S. N. Patnaik and R. H. Gallagher, 'Gradients of behaviour constraints and reanalysis via the integrated force method', *Int. j. numer. methods eng.*, **23**, 2205–2212 (1986).
37. S. N. Patnaik and N. K. Srivastava, 'On automated optimum design of trusses', *Comp. Methods Appl. Mech. Eng.*, 245–265 (1976).
38. G. N. Vanderplaats and F. Moses, 'Automated design of trusses for optimum geometry', *J. Struct. Div. Am. Soc. Civil Eng. Proc.*, **98**, 671–690 (1972).
39. L. Berke and N. S. Khot, *Use of Optimality Criteria Methods for Large Scale Systems*, Air Force Flight Dynamics Lab., Wright-Patterson Air Force Base, Ohio, 1974.
40. N. S. Khot, *Optimization of Structures for Strength and Stability Requirements*, Air Force Flight Dynamics Lab., Wright-Patterson Air Force Base, Ohio, 1973.
41. S. N. Patnaik, J. D. Guptill and L. Berke, 'Singularity in structural optimization', *Int. j. numer. methods eng.*, **36**, 931–944 (1993).
42. A. S. Gendy, J. D. Guptill, D. A. Hopkins and S. N. Patnaik, 'Large scale structural optimization with substructuring in a parallel computational environment', *OAI/OSC/NASA Proc. Symp. on Application of Parallel and Distributed Computing*, 1994, pp. 89–105.
43. S. N. Patnaik, R. M. Coroneos, J. D. Guptill and D. Hopkins, *Performance Trend of Different Algorithms for Structural Design Optimization*, NASA TM-4698, 1995.