EVALUATING MATHEMATICAL PROGRAMMING TECHNIQUES

Proceedings of a Conference Held at the
National Bureau of Standards, Boulder, CO

January 5-6, 1981

Edited by

John M. Mulvey
School of Engineering & Applied Science
Princeton University, Princeton, NJ 08594

This NBS special publication summarizes a two day conference that was held in Boulder, Colorado (January 5-6, 1981) to consider how mathematical programming techniques ought to be evaluated. Approximately fifty academic researchers, software developers and users participated (see Appendix A) in this first meeting devoted exclusively to computational testing of mathematical programs.

Following the introductory addresses, the remainder of the report is organized around nine topics:

1. Design and use of problem generators and hand selected test cases
2. Non-linear optimization codes and empirical tests .
3. Integer programming and combinatorics
4. Comparing three computational studies
5. Testing methodologies
6. Approaches to software evaluation from other disciplines
7. Special topics
8. Advances in network programs
9. On establishing a group for testing mathematical programs

Provided below is the table of contents and the abstracts of the papers which will appear in this proceedings. The abstracts appear in the order presented in the table of contents: Copies of individual papers can be obtained from the authors; requests for the proceedings should be directed to John Mulvey.

EVALUATING MATHEMATICAL PROGRAMMING TECHNIQUES

National Bureau of Standards Special Publication

(editor: John M. Mulvey)

3. Integer Programming and Combinatorics

- A computational comparison of five heuristic algorithms for the Euclidean traveling salesman problem
  (William R. Stewart, Jr.)

- Implementing an algorithm: performance considerations and a case study
  (Uwe Suhl)

- Which options provide the quickest solution
  (William J. Riley and Robert L. Sielken, Jr.)

4. A Comparison of Three Computational Studies

- Introduction
  (Ron S. Dembo)

- Remarks on the evaluation of nonlinear programming algorithms
  (David M. Himmelblau)

- Comments on evaluating algorithms and codes for mathematical programming
  (Robert B. Schnabel)

- Some comments on recent computational testing in mathematical programming
  (Jacques C.P. Bus)

- Remarks on the comparative experiments of Miele, Sandgren and Schittkowski
  (Ken M. Ragsdell)

5. Testing Methodologies

- In pursuit of a methodology for testing mathematical programming software
  (Karla L. Hoffman and Richard H.F. Jackson)

- Nonlinear programming methods with linear least square subproblems
  (Klaus Schittkowski)

- An outline for comparison testing of mathematical software -- illustrated by comparison testing of software which solves systems of nonlinear equations
  (Kathie L. Hiebert)

- A portable package for testing minimization algorithms
  (A. Buckley)

Every problem resp. code is described in a special chapter which is self-contained. Thus the reader will be able to apply a special routine without reading the whole book. All chapters are identically structured following the scheme

(a) Formulation of the problem
(b) Algorithm
(c) References
(d) Description of the program
(e) Program listings and sample output

In (a) we introduce the specific problem and state some alternative formulations thereof. In (b) the basic ideas of the implemented algorithm are roughly described. The mathematical background is only sketched. The reader can find details in the references, listed in section (c) of every chapter. In (d) we give directions for using the code. The reader will find here information on the structure of the program and a description of the INPUT and OUTPUT parameters with the respective dimension requirements. The total storage requirement and the running time for medium sized problems are given. The running time was determined either by testing a great variety of randomly generated problems or by standard problems from the literature on a CDC CYBER 76. In (e) the reader finds the complete list of the FORTRAN IV subroutines. These subroutines start with a comment where all parameters, arrays and variables are defined. A MAIN-program is added to describe the use of the subroutine and to enable the solution of the given sample problem.

All computer codes in this book are written in ANSI FORTRAN and they are therefore machine independent. The codes were developed and extensively tested on the CDC CYBER 76 of the Computer Center of the University of Cologne. Further tests were run on an IBM 4331 of the Sonderforschungsbereich 21 at the University of Bonn.

Reference: R. E. Burkard and U. Derigs: Assignment and Match Problems: Solution methods with FORTRAN-Programs, Lecture Notes in Economics and Mathematical Systems 184, SPRINGER, Berlin-Heidelberg-New York 1980.

REFERENCE

[1] G. Knolmayer, "Programmierungsmodelle fur die Produktions-programmplanung, Ein Beitrag zur Methodologie der Modellkonstruktion", Birkhauser-Verlag (Basel-boston-Stuttgart, 1980).

[2] G. Knolmayer, "Computational experiments in the formulation of linear product-mix and non-convex-production-investment models", submitted for publication.

[3] G. Knolmayer, "Computational experiments in the formulations of large-scale linear programs", in: Large-Scale Linear Programming, G.B. Dantzig, M.A.H. Dempster, M.J. Kallio ed., Vol. 2, IIASA Collaborative Proceedings Series, CP-81-S1 (Laxenburg, 1981), p. 865-887.

****

FORTRAN - PROGRAMS FOR ASSIGNMENT AND MATCHING PROBLEMS

Rainer E. Burkard and Ulrich Derigs
Mathematisches Institut — Industrieseminar
Universitat zu Koln — Universitat zu Koln
Weyertal 86 — Albertus-Magnus-Platz
5000 Koln 41 — 5000 Koln 41
Federal Republic of Germany — Federal Republic of Germany

Assignment and matching problems belong to those combinatorial optimization problems which are well understood in theory and have many applications in practice. Since a research group of the Mathematical Institute in Cologne has worked in this field for several years, we felt that a publication of the developed codes and algorithms would be useful, both for further research and academic tuition as well as for applications in practice.

A collection of FORTRAN-Programs for assignment and matching problems has recently been published by the authors of this note.

This book covers linear assignment problems with sum and bottleneck objective functions, cardinality matching problems, perfect matching problems with sum and bottleneck objective functions, the Chinese postman problem and optimal as well as heuristic algorithms for the quadratic assignment problem.

6. Approaches to Software Testing from Other Disciplines

   - Transportable test procedures for elementary function software
     (William J. Cody)
   - Testing and evaluation of statistical software
     (James E. Gentle)
   - TOOLPACK -- An integrated system of tools for mathematical software development
     (Leon J. Osterweil)
   - Overview of testing numerical software
     (Lloyd D. Fosdick)
   - The application of Halstead's software science difficulty measure to a set of programming projects
     (Charles P. Smith)

7. Special Topics

   - Mathematical programming algorithms in APL
     (Harlan Crowder)
   - A study of redundancy in mathematical programming
     (Mark Karwan, Vahid Lotfi, Jan Telgen, Stanley Zionts)

8. Advances in Networks

   - Solution strategies and algorithm behavior in large-scale network codes
     Richard S. Barr
   - Recursive piecewise-linear approximation methods for nonlinear networks
     (Robert R. Meyer)
   - Computational testing of assignment algorithms
     (Michael Engquist)

9. On Establishing a Group for Testing Mathematical Programs

   - Introduction
     (John M. Mulvey)
   - Participants

     Ron Dembo
     Richard Jackson
     Darwin Klingman
     Saul Gass
     Roy Marsten
     John Mulvey
     Ken Ragsdell
     Ronald Rardin

10. Appendix

- Conference Program
- List of participants
- A model for the performance evaluation in comparative studies
  (Klaus Schittkowski)
- Remarks on the comparative evaluation of algorithms
  for mathematical programming problems
  (Angelo Miele)
- Systematic approach for comparing the computational speed
  of unconstrained minimization algorithms
  (Salvador Gonzalez)
- The evaluation of optimization software for engineering design
  (Ken M. Ragsdell)

COMPARISONS OF EQUIVALENT FORMULATIONS IN MATHEMATICAL PROGRAMMING

Gerhard Knolmayer
University of Kiel
Institute of Business Administration
Olshausenstraße 40-60
D 2300 Kiel
FRG

Computational testing in mathematical programming has thus far concentrated on comparing optimization algorithms (codes). Another decision which influences the performance of a mathematical programming system is the way in which a given problem is formulated and submitted to the code. Often many equivalent formulations exist from which identical optimal activity levels can be derived (by using a report writer) and in which the optimal values of the objective function coincide. Therefore, the need for research on a methodology on formulating MP-models has been expressed.

Thus far, most studies in this area compare only two formulations. Reasons for this may be that the large number of equivalent formulations has not been fully recognized and/or that it is regarded cumbersome to produce these formulations. Therefore, an algorithm has been developed and programmed to generate eqivalent formulations for linear programming (LP) models which contain balance equations. These constraints with a RHS-element equal zero are typical for many important applications of linear programming, e.g. product-mix decisions. If a linear program contains K balance equations at least 2^K equivalent formulations exist.

For the computational study a rather sophisticated problem generator PPPGEN for LP-models of product-mix type was written in FORTRAN. Several parameters allow a very detailed specification of the model to be generated, e.g. for the number of options available in the technology.

The generated problem is transformed by eliminating balance equations via redefinitions of varibles. The program lists for the problem generator and the program which produces equivalent formulations are given in [1]. The resulting formulations are optimized by using APEX-III on a CYBER74. The results in [1] show that in the selection of a LP-formulation for a given problem one should not trust mathematical programming folklore which states that CPU-time rises roughly in proportion to the cube of the number of constraints; a far better explanation of computational effort is possible by taking the number of constraints as well as the number of nonzeros into account. Some of these results have been published in [3]. Additional results on the effects of eliminating balance equations in the "linear part" of a mixed integer model are discussed in [2]. For linear and mixed integer models, rules of thumb are developed to support the selection of appropriate formulations.

[MorGH81] J. J. Moré, B. S. Garbow, and K. E. Hillstrom, "Testing Unconstrained Optimization Software", ACM Trans. Math. Software 7 (1981), pp. 17-41.

[Sch79] L. Schrage, "A More Portable Fortran Random Number Generator", ACM Trans. Math. Software 5 (1979), pp. 132-138.

[Smi71] C. S. Smith, "Multiplicative Pseudo-Random Number Generators with Prime Modulus", J. Assoc. Comput. Mach. 18 (1971), pp. 586-593.

TEST PROBLEMS FOR COMPUTATIONAL EXPERIMENTS -- ISSUES AND TECHNIQUES

by

Ronald L. Rardin
Associate Professor, School of Industrial and System Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332.

and

Benjamin W. Lin
Assistant Professor, Department of Mechanical, Industrial and Aerospace Engineering, Rutgers University, Box 909, Piscataway, New Jersey 08854.

This paper compares and constrasts real world and random generated test problems as sources of standard tests for mathematical programming algorithms. Real problems are viewed as realizations from a test population of interest, and random problems are treated as models of the population. Methodolical advantages and difficulties inherent in the alternatives are highlighted, and methods for dealing with the limitations discussed.

THE DEFINITION AND GENERATION OF GEOMETRICALLY RANDOM LINEAR CONSTRAINTS SETS

by

Jerrold H. May
Graduate School of Business
Unversity of Pittsburgh
Pittsburgh, PA 15260

and

Robert L. Smith
Department of Industrial and Operations Engineering
University of Michigan
Ann Arbor, MI 48109

The conventional procedure for generating random linear constraint sets independently and uniformly is one where one selects the constraint coefficients from some psudo-random number generation procedure. Structure is usually imposed through some kind of rejection technique. Recent work of Van Dam and Telgen indicates that this type of generator tends to produce geometrically atypical polytopes. We define and construct a generator that produces geometrically random constraint sets; that is, their probability measure is invariant to the choice of coordinate system used. Moreover, an extremely efficient technique is presented for making an unbiased selection of a feasible polytope. Conventional approaches often guarantee feasibility by implicitly selecting that randomly generated polytope covering the origin. Such a procedure biases the selection in favor of large polytopes.

NETGEN-II: A SYSTEM FOR GENERATING STRUCTURED NETWORK-BASED MATHEMATICAL PROGRAMMING TEST PROBLEMS

by

Joyce J. Elam
Darwin Klingman
College of Business Administration
University of Texas
Austin, Texas 78712

The increased importance of designing and implementing algorithms to solve particular management problems has created the need for more robust test problem generators that can match the overall strucuture and parameter values of these problems. Of particular interest are management problems that can be modeled using a network structure. This paper discusses the design of a system for generating network-based mathematical programming test problems that conform to user-supplied structural and parameter characteristics.

CONSTRUCTION OF NONLINEAR PROGRAMMING TEST PROBLEMS WITH KNOWN SOLUTION CHARACTERISTICS

by

Gideon Lidor
Department of Computer Sciences, The City College
New York, N.Y. 10031

The ability to supply realistic-looking test problems with known characteristics is essential for testing the efficiency and robustness of nonlinear programming algorithms and codes. This paper deals with methods of constructing (random) nonlinear programming test problems with known solutions and specified characteristics which usually affect the success of algorithms and their related software. In particular, it is shown how test problems with given multiple stationary points can be constructed, allowing for programs with several local optima and thus providing for more rigourous testing.

REFERENCES

[CroDM79] H. Crowder, R. S. Dembo, and J. M. Mulvey, "On Reporting Computational Experiments with Mathematical Software", ACM Trans. Math. Software 5 (1979), pp. 193-203.

[DenGW80] J. E. Dennis, Jr., D. M. Gay, and R. E. Welsch, "An Adaptive Nonlinear Least-Squares Algorithm", Tech. Rep. #TR-20, Center for Computational Research in Economics and Management Science, Mass. Inst. of Technology, 1980. To appear in ACM Trans. Math. Software.

[Fos79] Performance Evaluation of Numerical Software, edited by L. D. Fosdick, North-Holland, Amsterdam, New York and Oxford, 1979.

[Gay80] D. M. Gay, "Subroutines for General Unconstrained Minimization Using a Model/Trust-Region Approach", Tech. Rep. #TR-18, Center for Computational Research in Economics and Management Science, Mass. Inst. of Technology, 1980.

[Gay81] D. M. Gay, "Solving Interval Linear Equations", Tech. Rep. #TR-24, Center for Computational Research in Economics and Management Science, Mass. Inst. of Technology, 1981.

[GayS78] D. M. Gay and R. B. Schnabel, "Solving Systems of Nonlinear Equations by Broyden's Method with Projected Updates", pp. 245-281 of Nonlinear Programming 3, edited by O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, Academic Press, New York, 1978.

[GilM78] P. E. Gill and W. Murray, "Algorithm for the Solution of the Nonlinear Least-Squares Problem", SIAM J. Numer. Anal. 15 (1978), pp. 977-992.

[Hoa76] D. C. Hoaglin, "Theoretical Properties of Congruential Random-Number Generators: An Empirical View", Memorandum NS-340, Dept. of Statistics, Harvard Univ., 1976.

[Knu69] D. E. Knuth, The Art of Computer Programming, vol. 2: Seminumerical Algorithms, Addison-Wesley, Reading, MA, 1969.

COMPARING ALGORITHMIC VARIATIONS

	DEFAULT			NOIMODSW			NOINTDBL			NOGRDTST		
	NF	NG	NS	NF	NG	NS	NF	NG	NS	NF	NG	NS
R = 0	1.12	1.06	22	1.10	1.03	22	1.15	1.27	22	1.15	1.08	22
R ≠ 0	1.09	1.02	28	1.15	1.05	27	1.05	1.18	28	1.18	1.09	27
Both	1.10	1.04	50	1.13	1.04	49	1.10	1.22	50	1.17	1.09	49

Table IV: Mean Weighted NF and NG values for Table II

rithmic variations considered do not make a great deal of difference. The data in Table II were included in [DenGW80] in response to a referee's question about why we chose the particular variations labeled DEFAULT.)

Concluding Remarks

Uncertainties about what is the most appropriate test problem space and hence about how to weigh the results of computational tests leave most comparative testing open to some criticism. This is certainly true in the approach to comparing algorithmic variations described above, in part because different test problem spaces and ways to weigh results are doubtless most appropriate to different applications. Nonetheless, the approach described above is the best one I know at the time of this writing.

A COMPARISON OF REAL-WORLD LINEAR PROGRAMS AND THEIR RANDOMLY-GENERATED ANALOGS

by

Richard P. O'Neill, Director,
Oil and Gas Analysis Division, EI-522, Rm 4520
MS 4530, 12th & Pennsylvania Avenue, N.W.
Washington, D.C. 20461

The intent of the study is to determine the ability of randomly-generated problems to simulate real-world problems for the purposes of testing, benchmarking and comparing software implementations of solution algorithms, and to determine if the "degree of randomness" is related to the difficulty in obtaining a solution. Randomly-generated analogs are defined to be problems created with some of the characteristics of a real-world (actual application) problem, but containing data with random elements. These analogs fall into classes that can be characterized by "nearness" to the real-world problem. The second class is obtained by randomizing the ones in the Boolean image of the problem data. The third class consists of problems obtained from software generator that accepts the problem characteristics as input. The random elements are generated from uniform and normal distributions.

The measures of difficulty include central processor time, iterations and bumps in the optimal basis. Several optimizers are used in the study. In general, the results show that the difficulty increases with the "degree of randomness" and difficulty difference increases as a function of size.

EVIDENCE OF FUNDAMENTAL DIFFICULTIES IN NONLINEAR OPTIMIZATION CODE COMPARISONS

by

Earnest D. Eason
Failure Analysis Associate
750 Welch Road, Suite 116
Palo Alto, California 94304

Several nonlinear optimization code comparisons have been published, providing data for picking a code for a given application or developing new codes. The common performance measures (number of function evaluations, standardized computer time, number of problems solved) are intuitively machine independent, which encourages such use. Unfortunately, the relative performance of optimization codes does depend on the computer and compiler used for testing, and this dependence is evident regardless of the performance measure. In additon, relative performance measured on a single machine may depend significantly on the desired degree of accuracy, the choice of test problem(s), the chosen performance measure, and even the time of day (machine workload) when the test are run. Numerical evidence of these difficulties is presented, based on test of the same problem and algorithm decks on several different computers, with various compilers, problem sets, accuracy levels, and performance measures.

A STATISTICAL REVIEW OF THE SANDGREN-RAGSDELL COMPARATIVE STUDY

by

Eric Sandgren, Staff Engineer
IBM Corporation, Information Systems Division
Lexington, Kentucky

A statistical analysis of the solution times of the algorithms in the Sandgren-Ragsdell study is conducted. An analysis of variance is performed to demonstrate that there is statistical evidence that selected codes are superior to others on the basis of their relative solution times. A logarithmic transformation is used to produce a semi-normal distribution of the solution times with a variance assumed to be equal for all of the algorithms. A paired comparison is then conducted on the differences in the mean logarithmic solution times for each of the algorithms over the entire test problem set. The selected confidence level for all comparisons was fixed at 95%. The factors contributing to the success of this analysis are discussed as well as the additional data which would be required to conduct this type of analysis in general.

function evaluations for each of three algorithmic variations on a small set of test problems. It also gives corresponding normalized function evaluation counts obtained by dividing the raw function evaluation count by the minimum count (over the three variations) for the test problem in question. Finally, it reports the grand means (and standard deviations) of these normalized counts for each algorithmic variation; these grand means are the overall weighted costs mentioned above (with a weight of unity for each successfully solved test problem considered).

The situation is more complicated when there are several performance indicators. In this case it sometimes seems reasonable to compute a separate overall weighted summary for each indicator. Consider, for example, Table II. Here I think it most appropriate to compute separate weighted sums for the NF and NG columns. Because two distinct kinds of problems are being considered, namely so-called "zero-residual" and "nonzero-residual" problems, it seems reasonable to compute weighted sums over each of these problem classes as well as weighted sums over all the problems. Proceeding as in Table III (and counting entries of A, B, R, and S in the C columns of Table II as successes), I obtain Table IV. Entries in the columns labeled NS are the number of successfully solved problems in the relevant class; "R = O" refers to the zero-residual problems, "R ≠ O" to the nonzero-residual problems, and "Both" to the whole set of test problems. (Table IV suggests that the algo-

COMPARING ALGORITHMIC VARIATIONS

	Total Function Evaluations			Normalized Function Evaluations		
Problem	Broyden's "Good"	Algorithm II $\tau = 10$	Algorithm II $\tau = 100$	Broyden's "Good"	Algorithm II $\tau = 10$	Algorithm II $\tau = 100$
1.5	31	27	28	1.15	1.00	1.04
2.2	11	10	10	1.10	1.00	1.00
3.2	9	9	9	1.00	1.00	1.00
3.3	13	11	13	1.18	1.00	1.18
3.4	19	23	23	1.00	1.21	1.21
3.5	20	24	23	1.00	1.20	1.15
3.6	$(31)^1$	26	33	--	1.00	1.27
3.7	45	35	36	1.29	1.00	1.03
4.2	12	10	10	1.20	1.00	1.00
5.3	15	$(28)^1$	$(28)^1$	1.00	--	--
5.3^2	16	15	15	1.07	1.00	1.00
6.6^3	62	29	60	2.14	1.00	2.07
6.6^2	32	28	57	1.14	1.00	2.04
7.5	13	13	13	1.00	1.00	1.00
7.10	21	20	20	1.05	1.00	1.00
	Function Evaluations Required to Achieve $\|F\| < 10^{-10}$		Mean	1.17	1.03	1.21
			Std.Dev.	.29	.074	.37
			Failures	1	1	1

Notes:

1. Broyden's [1965] quadratic interpolation technique failed to reduce $\|F\|$ in 10 function evaluations. The number reported is the total number of function evaluations at the time of failure.
2. $\|F\|$ was allowed to increase as much as twofold (per step) and a maximum steplength of 10 rather than 1 was allowed.
3. A maximum steplength $\|s_i\|_\infty$ of 10 rather than 1 was allowed.

Table III: Example of a Simple Weighting Method

A METHODOLGOICAL APPROACH TO TESTING OF NLP-SOFTWARE

by

Jacques C. P. Bus,
Mathematical Centre,
Amsterdam

Prior to any actual testing of software there should be:

- knowledge of the software and their underlying algorithms;
- knowledge of the class of problems to be solved by this software
- goals of testing.

We present a methodological setup for testing nonlinear programming software which is based on such a priori knowledge. This leads to discussion of topics like:

- representivity of testproblems;
- use of testproblem generators;
- performance measures;
- presentation of testresults;
- presentation of final conclusions.

These ideas will be illustrated with a project which has been performed to evaluate a set of Newton-like methods for solving systems of nonlinear equations.

A COMPUTATIONAL COMPARISON OF FIVE HEURISTIC ALGORITHMS FOR THE EUCLIDEAN TRAVELING SALESMAN PROBLEM

by

William R. Stewart, Jr.
School of Business, College of William and Mary
Williamsburg, VA 23185

This paper presents a computation comparison of five heuristic algorithms for the traveling salesman problem (TSP). Each of these algorithms is a composite algorithm consisting of a simple tour construction algorithm and a post-processor. The five tour construction procedures considered are all insertion algorithms which may be easily implemented. The post-processor used in the five composite algorithms is a modified 3-opt procedure that is shown to outperform both the 3-opt and 2-opt branch exchange procedures. The final result of the computational tests is the conclusion that there are simple and easily implemented heuristic procedures that will produce high quality solutions to the TSP in a moderate amount of computer time.

IMPLEMENTING AN ALGORITHM: PERFORMANCE CONSIDERATIONS AND A CASE STUDY

by

Uwe Suhl
Freie Universitat Berlin
Fachbereich Wirtschaftswissenschaft
Fachrichtung Wirtschaftsinformatik
Garysta3e 21, D-1000 Berlin 33

After a general discussion of program performance and its influence factors we consider a case study. A specific version of TOYODA's heuristic algorithms for solving multidimensional knapsack problems was implemented in three different FORTRAN programs. Starting from a straightforward program, we introduced increasingly sophisticated data structures to improve the asymptotic computational time complexity of the programs. All programs were compiled with the enhanced version of the FORTRAN H compiler available on IBM /370, 43xx, 303x machines, using its four compiler options. A computational study was performed with a series of randomly generated test problems with up to 3200 variables and constraints and about 70000 nonzeros. The purpose of this study was to measure virtual central processor times and working set sizes as a function of problem size and to study the influence of code optimization performed by the compiler. Since all three programs are representations of the same mathematical algorithm, conclusions can be drawn as to the relative importance of the influence factors on program performance. As it is also demonstrated the potential performance of a representation of an abstract algorithm may be difficult to project without a careful implementation.

WHICH OPTIONS PROVIDE THE QUICKEST SOLUTION

by

William J. Riley
Robert L. Sielken, Jr.
Institute of Statisitcs
College Station, University of Texas
Houston, TX 77843

Frequently algorithm users can select their solution strategy by choosing from among various options for each of several algorithm factors. If the algorithm will always eventually find a solution, the important question is which combination of options is most likely to be "best". A general statistical approach to answering this question is illustrated in the context of a new integer linear programming algorithm where "best" is quickest.

The integer programming algorithm is a sophisticated implicit enumeration algorithm. The four factors where the user must select an option are (1) augmenting partial solutions, (2) backtracking, (3) fathoming on the basis of binary feasibility and optimality indicators, and (4) use of linear programming on the relaxed problem which includes penalties, cuts, surrogate constraints, and associated fathoming. There are several options per factor so that the algorithm can function in over 14, 000 different modes.

COMPARING ALGORITHMIC VARIATIONS

		DEFAULT			NOIMODSW			NOINTDBL			NOGRDTST			
PROBLEM	LS	NF	NC	C	NF	NG	C	NF	NG	C	NF	NG	C	NOTE
ROSNBROK	0	26	19	A	27	19	A	26	21	X	21	18	A	
ROSNBROK	1	57	39	A	36	29	A	74	57	A	45	35	A	
ROSNBROK	2	141	121	A	135	115	A	164	145	A	210	155	A	
HELIX	0	13	11	A	13	11	A	13	11	A	17	14	X	
HELIX	1	19	16	A	17	14	A	19	16	A	18	15	X	
HELIX	2	103	45	F	110	33	F	20	16	X	99	43	F	
SINGULAR	0	20	20	A	20	20	A	20	20	A	20	20	A	
SINGULAR	1	26	25	A	26	25	A	25	25	A	28	25	A	
SINGULAR	2	34	27	A	34	27	A	31	31	A	34	27	A	
WOODS	0	70	47	A	65	47	X	59	51	A	75	49	X	
WOODS	1	59	46	A	71	48	X	41	37	A	54	42	A	
WOODS	2	77	53	X	77	53	X	59	55	X	87	54	A	
ZANGWILL	0	3	3	A	3	3	A	3	3	A	3	3	A	
ENGVALL	0	17	13	X	16	13	X	17	14	X	17	13	X	
ENGVALL	1	21	19	X	23	19	X	16	16	X	22	19	A	
ENGVALL	2	31	26	A	31	26	A	36	34	X	36	28	A	
BRANIN	0	2	2	A	2	2	A	2	2	A	2	2	A	
BRANIN	1	18	15	A	18	15	A	25	25	A	16	13	A	
BRANIN	2	20	10	A	20	10	A	39	38	A	23	12	A	
BEALE	0	10	9	A	11	9	A	10	9	A	10	9	A	
BEALE	1	6	6	A	6	6	A	6	6	A	6	6	A	
CRAGG	0	24	23	A	23	21	A	24	23	A	25	24	A	
CRAGG	1	80	47	R	80	47	R	109	93	B	80	49	B	1
BOX	0	7	7	X	7	7	X	7	7	X	7	7	X	
BOX	1	16	10	S	17	10	S	16	10	S	15	11	R	
DAVIDON1	0	20	15	X	20	15	X	16	16	I	20	13	E	
FRDSTEIN	0	9	8	R	8	7	R	9	8	R	9	8	R	
FRDSTEIN	1	18	13	R	18	13	R	19	17	R	18	14	B	
FRDSTEIN	2	28	19	R	35	22	B	29	27	R	29	20	B	
WATSON6	0	12	10	B	12	10	B	11	10	B	12	10	B	
WATSON9	0	10	9	R	10	9	R	10	9	B	11	9	B	
WATSON12	0	14	12	R	14	12	R	14	13	R	18	13	B	
WATSON20	0	18	16	I	18	16	I	16	16	I	20	15	E	2
CHEBQD8	0	23	18	B	24	19	B	19	14	R	23	18	B	
CHEBQD8	1	77	57	R	118	76	R	112	98	R	104	82	R	
BROWN	0	18	17	R	18	17	R	20	19	R	20	18	R	
BROWN	1	22	16	R	25	19	R	24	23	R	26	20	B	
BROWN	2	31	21	R	32	22	B	30	29	B	32	23	R	
BARD	0	7	7	R	7	7	R	7	7	R	7	7	R	
BARD	1	32	23	S	32	23	S	29	29	S	32	23	S	
BARD	2	70	28	R	77	28	R	66	43	R	66	30	R	
JENNRICH	0	15	13	R	15	13	R	15	13	R	15	13	R	
KOWALIK	0	11	10	R	13	10	B	11	10	R	13	10	R	
KOWALIK	1	130	75	S	244	100	F	109	84	S	124	76	S	3
KOWALIK	2	75	58	R	74	58	R	78	62	R	117	81	R	
OSBORNE1	0	27	22	R	31	22	R	28	23	R	34	23	R	
OSBORNE2	0	17	16	B	17	16	B	17	16	B	18	16	B	
OSBORNE2	1	26	12	S	27	12	S	13	12	S	26	12	S	
MADSEN	0	12	12	R	12	12	R	12	12	R	12	12	R	
MADSEN	1	16	15	R	16	15	R	18	18	B	19	17	R	
MADSEN	2	28	20	R	29	21	R	25	25	R	27	22	R	
MEYER	0	335	206	X	343	214	B	209	181	B	351	213	X	

Table II: Summaries of Several Algorithmic Variations

Entries in the NOTE column above refer to the Notes on Table IV in [DenGW80].

tions found the same solution). To simultaneously compare the work expended by several algorithmic variations in solving the test problems, I sometimes find it helpful to produce a one-page summary of the relevant performance indicators. Table II (which is Table IV of [DenGW80]) provides an example of this; the columns labeled NF, NG, and C give the numbers of function and gradient evaluations and return code for each of four variations. Looking such a table over, and knowing something about the various test problems, one can draw a subjective conclusion about the merits of the competing algorihmic variations.

The above method of analysis is an informal one; it is the one that I apply first when in the midst of some work. Sometimes it seems desirable to formalize it by computing some kind of overall weighted cost or costs for the execution of each algorithmic variation on the set of test problems. The best way to weight the various test problems is not at all clear; because of this, when writing reports I generally prefer to present the raw data, so that readers can decide whether they agree with my conclusions.

Table III (which is Table I of [GayS78]) illustrates a weighting technique that I have sometimes used. This table is concerned with an algorithm for solving systems of nonlinear equations. In this simple case we thought it reasonable to measure computational cost by counting the number of function evaluations (i.e., evaluations of the entire set of equations being solved). Table III gives the number of

A statisitical evaluation of the average effects of each option on the algorithm's speed and the interactions between options is obtained using analysis of variance techniques. The design of the experiment, the linear model, and the analysis of the resulting data are discussed. The generality of this approach to analyzing algorithm components is emphasized.

COMPARATIVE COMPUTATIONAL STUDIES IN MATHEMATICAL PROGRAMMING

by

Ron S. Dembo
Yale School of Organization and Management
Box 1A
New Haven, CT 06520

This session is concerned with identifying ideas that will guide future studies on comparisons of algorithms and codes. To do so we have selected three recent computational studies with differing view points. The authors of these studies will make a brief presentation outlining the methodology they used, their views on the good and bad points in the study, and their suggestions for future studies. Following these presentations members of the panel, who have been selected so as to cover a wide range of opinions on the subject, will comment on the points that have been made. The discussion will then be open to the general audience.

IN PURSUIT OF A METHODOLOGY FOR TESTING MATHEMATICAL PROGRAMMING SOFTWARE

by

K. L. Hoffman
Richard H. F. Jackson
Center for Applied Mathematics
National Bureau of Standards
Washington, D.C. 20234

The resolve the often conflicting and confusing results of computational testing of mathematical software reported in the literature, the Committee on Algorithms of the Mathematical Programming Society has developed guidelines which present minimum standards to which all papers reporting such effort should conform. Although guidelines now exist, the development of sound methodologies for this testing is at the embryonic stage. This paper surveys the results to date of computational test efforts in each of the major fields of mathematical programming and reviews to what extent these efforts have advanced the goal of developing methodologies for testing MP software. Directions for future research are presented.

NONLINEAR PROGRAMMING METHODS WITH LINEAR LEAST SQUARES SUBPROBLEMS

by

Klaus Schittkowski
Institute for Angewandte Mathematik und Statistik
Universitut Wurzburg
8702 Wurzburg WEST GERMANY

This paper presents the results of an extensive comparative study of nonlinear optimization algorithms, cf. [8]. This study indicates that quadratic approximation methods which are characterized by solving a sequence of quadratic subproblems recursively, belong to the most efficient and reliable nonlinear programming algorithms available at present. The purpose of the paper is to investigate their numerical performance in more detail. In particular, the dependence of the overall performance on alternative quadratic subproblem strategics is tested, and the paper indicates how the efficiency of quadratic approximation method can be improved.

AN OUTLINE FOR COMPARISON TESTING OF MATHEMATICAL SOFTWARE ILLUSTRATED BY COMPARISON TESTINGS OF SOFTWARE WHICH SOLVES SYSTEMS OF NONLINEAR EQUATIONS

by

K. L. Hiebert
Sandia National Laboratories
Albuquerque, New Mexico 87185

This paper evaluates information collected during comparison testing of mathematical software for solving systems of nonlinear equations. Of the two methods being surveyed, one solves nonlinear least squares problems while the other solves square systems of equations. An outline for comparison testing is illustrated with examples from the two methods, and the difficulties in mathematical software testing are discussed.

COMPARING ALGORITHMIC VARIATIONS

PROBLEM	N	P	ITER	NF	NG	C	XOSCAL	FINAL F	PRELDF	NPRELDF	RELDX
ROSNBROK	2	2	18	26	19	A	1.	0.973E-32	0.100E+01	0.100E+01	0.818E-03
ROSNBROK	2	2	38	57	39	A	10.	0.973E-32	0.100E+01	0.100E+01	0.594E-04
ROSNBROK	2	2	120	141	121	A	100.	0.973E-32	0.100E+01	0.100E+01	0.440E-03
HELIX	3	3	10	13	11	A	1.	0.276E-20	0.100E+01	0.100E+01	0.145E-05
HELIX	3	3	15	19	16	A	10.	0.120E-20	0.100E+01	0.100E+01	0.244E-05
HELIX	3	3	45	103	45	F	100.	0.120E+05	0.135E-12	-0.984E+00	0.181E-13
SINGULAR	4	4	19	20	20	A	1.	0.107E-20	0.100E+01	0.100E+01	0.333E+00
SINGULAR	4	4	24	26	25	A	10.	0.751E-21	0.100E+01	0.100E+01	0.333E+00
SINGULAR	4	4	26	34	27	A	100.	0.224E-20	0.100E+01	0.100E+01	0.333E+00
WOODS	7	4	46	70	47	A	1.	0.232E-23	0.100E+01	0.100E+01	0.197E-06
WOODS	7	4	45	59	46	A	10.	0.487E-26	0.100E+01	0.100E+01	0.426E-07
WOODS	7	4	52	77	53	X	100.	0.0	0.100E+01	0.100E+01	0.359E-10
ZANGWILL	3	3	2	3	3	A	1.	0.426E-27	0.100E+01	0.100E+01	0.100E+01
ENGVALL	5	3	12	17	13	X	1.	0.279E-32	0.100E+01	0.100E+01	0.357E-10
ENGVALL	5	3	18	21	19	X	10.	0.631E-29	0.100E+01	0.100E+01	0.268E-08
ENGVALL	5	3	25	31	26	A	100.	0.164E-22	0.100E+01	0.100E+01	0.107E-06
BRANIN	2	2	1	2	2	A	1.	0.162E-28	0.100E+01	0.100E+01	0.100E+01
BRANIN	2	2	14	18	15	A	10.	0.662E-29	0.100E+01	0.100E+01	0.100E+01
BRANIN	2	2	9	20	10	A	100.	0.138E-20	0.100E+01	0.100E+01	0.100E+01
BEALE	3	2	8	10	9	A	1.	0.893E-26	0.100E+01	0.100E+01	0.116E-06
BEALE	3	2	5	6	6	A	10.	0.148E-21	0.100E+01	0.100E+01	0.115E-05
CRAGG	5	4	22	24	23	A	1.	0.217E-20	0.100E+01	0.100E+01	0.253E-07
CRAGG	5	4	46	80	47	R	10.	0.617E+05	0.919E-11	0.919E-11	0.979E-07
BOX	10	3	6	7	7	X	1.	0.174E-31	0.100E+01	0.100E+01	0.196E-09
BOX	10	3	10	16	10	S	10.	0.378E-01	0.780E-17	-0.537E-10	0.177E-13
DAVIDON1	15	15	14	20	15	X	1.	0.400E-18	0.100E+01	0.100E+01	0.105E-06
FRDSTEIN	2	2	7	9	8	R	1.	0.245E+02	0.359E-11	0.359E-11	0.769E-06
FRDSTEIN	2	2	12	18	13	R	10.	0.245E+02	0.532E-13	0.532E-13	0.934E-07
FRDSTEIN	2	2	18	28	19	R	100.	0.245E+02	0.782E-13	0.782E-13	0.418E-07
WATSON6	31	6	10	12	10	B	1.	0.114E-02	0.422E-19	0.422E-19	0.142E-10
WATSON9	31	9	8	10	9	R	1.	0.700E-06	0.173E-10	0.173E-10	0.360E-07
WATSON12	31	12	12	14	12	R	1.	0.236E-09	0.122E-13	0.122E-13	0.254E-07
WATSON20	31	20	15	18	16	I	1.	0.651E-17	0.532E+00	0.999E+00	0.270E+00
CHEBQD8	8	8	17	23	18	B	1.	0.176E-02	0.277E-11	0.277E-11	0.103E-07
CHEBQD8	8	8	56	77	57	R	10.	0.176E-02	0.392E-10	0.392E-10	0.841E-07
BROWN	20	4	16	18	17	R	1.	0.429E+05	0.224E-10	0.224E-10	0.228E-06
BROWN	20	4	15	22	16	R	10.	0.429E+05	0.848E-12	0.848E-12	0.696E-07
BROWN	20	4	20	31	21	R	100.	0.429E+05	0.111E-10	0.111E-10	0.187E-06
BARD	15	3	6	7	7	R	1.	0.411E-02	0.270E-12	0.270E-12	0.119E-06
BARD	15	3	22	32	23	S	10.	0.871E+01	0.896E-06	-0.396E-10	0.243E+00
BARD	15	3	27	70	28	R	100.	0.411E-02	0.411E-10	0.411E-10	0.146E-05
JENNRICH	10	2	12	15	13	R	1.	0.622E+02	0.169E-12	0.169E-12	0.134E-06
KOWALIK	11	4	9	11	10	R	1.	0.154E-03	0.421E-10	0.421E-10	0.423E-06
KOWALIK	11	4	74	130	75	S	10.	0.514E-03	0.209E-06	-0.689E-10	0.242E+00
KOWALIK	11	4	57	75	58	R	100.	0.154E-03	0.470E-11	0.470E-11	0.103E-06
OSBORNE1	33	5	21	27	22	R	1.	0.273E-04	0.332E-11	0.332E-11	0.524E-06
OSBORNE2	65	11	15	17	16	B	1.	0.201E-01	0.492E-12	0.492E-12	0.933E-08
OSBORNE2	65	11	11	26	12	S	10.	0.895E+00	0.353E-10	-0.353E-10	0.879E-07
MADSEN	3	2	11	12	12	R	1.	0.387E+00	0.105E-13	0.105E-13	0.496E-07
MADSEN	3	2	14	16	15	R	10.	0.387E+00	0.154E-11	0.154E-11	0.584E-06
MADSEN	3	2	19	28	20	R	100.	0.387E+00	0.120E-12	0.120E-12	0.152E-06
MEYER	16	3	205	335	206	X	1.	0.440E+02	0.705E-05	0.705E-05	0.136E-07

Table I: Sample Performance Indicator Summary Page

nent of the computational effort, e.g. the number of iterations, function evaluations, and gradient evaluations.

4. Running the tests

It seems best to automate the testing procedure as much as possible. I generally arrange things so that by issuing a small number of commands, I can test several algorithmic variations on a battery of test problems. It is very helpful to save all possibly interesting performance indicators in machine-readable form for later reference.

5. Analyzing the test results

For each algorithmic variation considered, I find it helpful to produce a page of test results having one line of performance indicators per test problem. Table I below (from which Table II of [DenGW80] is extracted) provides an example of this: the columns labeled PROBLEM, N, P, and XOSCAL describe the test problem, and the remaining columns contain performance indicators. Glancing at the columns labeled C, FINAL F, PRELDF, NPRELDF, and RELDX is usually enough to tell me which solution was found and whether something unusual happened; occasionally I must go back to the computer for more details. This exercise helps me to compare algorithmic variations, since it gives information on reliability and tells me for which problems it is most reasonable to compare the computational effort expended in finding a solution (namely the problems on which both varia-

A PORTABLE PACKAGE FOR TESTING MINIMIZATION ALGORITHMS

by

A. Buckley
Mathematics Department
Concordia University
7141 Sherbrooke St
W. Montreal Canada H4B.1R6

A package of routines designed to aid in the process of testing minimization (or other types of) algorithms will be discussed. The intention is to provide a vehicle for evaluating codes which is portable, versatile and simple to use. Ease of use should encourage algorithm designers to use the package, versatility will ensure that it is applicable to testing many different routines, and portability will mean that it is available to anyone. This last point should contribute to consistency in the testing of algorithms. The emphasis is not on actual testing, but is on the presentation of some ideas of how one can design a useful testing mechanism.

TRANSPORTABLE TEST PROCEDURES FOR ELEMENTARY FUNCTION SOFTWARE

by

W. J. Cody
Applied Mathematics Division
Argonne National Laboratory
Argonee, Illinois 60439

This paper explores the principles and numerics behind the use of identities to test the accuracy of elementary function software. The success of this approach depends critically on proper matching of identity and test interval with the purpose of the test, and on careful implementation based on an understanding of computer arithmetic systems and inherent accuracy limitation for computation software.

TESTING AND EVALUATION OF STATISTICAL SOFTWARE

by

James E. Gentle
IMSL, Inc.
7500 Bellaire
Houston, TX 77036

Data sets analyzed by statisticians are likely to be ill-conditioned for the kinds of computations ordinarily performed on them. For this reason, most of the activity in testing and validation of statistical software has centered on numerical error analysis. The most generally effective testing and validation method has been test data generation. Once the parameters of numerical condition are identified, methods for systematically generating test data sets can be developed. In the case of least squares computations, for example, collinearity and stiffness seem to be the major components of condition. Test data sets with prespecified collinearity and stiffness are discussed.

The problem of software validation goes beyond the tests of the kernels that perform numerical computations. A program may involve the use of several computational modules; and errors in the program often occur in the setup stages in moving from one computational module to another. A stochastic model for errors remaining in a program after a sequence of tests is presented and discussed.

There are many statistical computations for which perturbation methods can be used easily to assess correctness. Perturbation methods can be employed by the user so as to avoid the bigger question of testing the software; the test is for the performance of the software on the specific problem of interest.

TOOLPACK-AN INTEGRATED SYSTEM OF TOOLS FOR MATHEMATICAL SOFTWARE DEVELOPMENT

by

Leon Osterweil
Department of Computer Science
University of Colorado,
Boulder, CO 80309

This paper describes the approach being taken in configuring a set of tool capabilities whose goal is the support of mathematical software development. The TOOLPACK tool set is being designed to support such activities as editing, testing, analysis, formatting, transformation, documentation and transporting of code. Tools for realizing most of these functional capabilities already exist, yet TOOLPACK aims to do far more than simply bring them together as a collection of side-by-side individual tools. TOOLPACK seeks to merge these capabilities into a system which is smoothly integrated both internally and from a users's external point of view. The internal integration approach involves the decomposition of all tools into a more or less standard set of

problems have several solutions, e.g. multiple local minima in the case of unconstrained minimization. So long as an algorithm finds some solution, I generally regard it to have succeeded, but when different algorithms (or algorithmic variations) find different solutions, I like to record some information about the solutions found (e.g. the final objective function value) for possible later use in a subjective comparison of the performance of the algorithms.

There are a number of components to the cost of running a computer code, such as the amount of memory and CPU time used and the amount of input/output performed. The algorithmic variations that I study usually require about the same amount of memory and usually do not involve explicit input/output (aside from optional printing), so in effect I am primarily concerned with measuring computational effort (CPU time). The codes I work with usually call one or two user-supplied, problem-dependent subroutines, e.g. to compute the objective function value and gradient, so there are several components to the computational effort: the time spent in the user-supplied subroutines and the time spent in performing the rest of the algorithm, i.e., the algorithmic overhead time. To allow rough extrapolations to other problem complexities (e.g. problems where the user-supplied subroutines require considerably longer to run than do the test-problem subroutines) and to other computing environments, I advocate a practice that finally is becoming more commonplace: keeping a separate count for each major compo-

COMPARING ALGORITHMIC VARIATIONS

Let us digress briefly to consider portable random-number generators. In this regard, I recommend reading [Sch79]. For problems where a "course" generator suffices, I recommend the simple generator used in [Gay81]. It is one that Virginia Klema and I learned about from David Hoaglin; it comes from [Smi71], and it passes the spectral test with flying colors -- see [Hoa76] and [Knu69]. It determines a pseudo-random sequence $i_1, i_2, \cdots$ of integers between 1 and 9972 from the formula

$$i_{k+1} = (3432 \cdot i_k) \bmod 9973.$$

If necessary, this may be computed as

$$i_{k+1} = [52 \cdot \{(66 \cdot i_k) \bmod 9973\}] \bmod 9973,$$

which requires only 20 bits of precision.

3. Choosing performance indicators

When deciding which of several computer codes to use, one is generally interested in asking how reliable (robust) the various codes are, how easy they are to use, and how costly they are to run. Ease of use is generally not an issue in my testing, but I am interested in the effects that algorithmic variations have on the reliability and cost of using the basic algorithm under study. By reliability I mean, roughly speaking, the ability to solve the given test problems after a reasonable amount of computational effort. A simple indicator thereof is a count of the number of successes and failures. The matter of deciding what is a success and what is a failure is an issue here. Many test

modular "tool fragments." The external integration approach involves the creation of a command language and a set of conceptual entities which is close to the conceptual set used by mathematical software writers in the process of creating their software. This paper describes both of these integration approaches as well as the rather considerable and novel software support needed to make them work.

OVERVIEW OF TESTING OF NUMERICAL SOFTWARE

by

Lloyd d. Fosdick
Department of Computer Science
University of Colorado, Boulder, CO 80309

The purposes of program testing are to expose errors; to insure standands for portability, robustness, etc. are met; and to evaluate performance. A selection of work in these three applications of program testing is presented here.

A variety of tools and techniques for exposing errors in programs have been proposed. They range from measurements of testing thoroughness, to models for predicting errors, to attempts to prove a limited form of correctness. For various reasons, including cost, availability of tools, and interpretation of results, testing of numerical software for the purpose of exposing errors has made only limited use of these tools and techniques.

The problem of insuring standards are met is easier to deal with, and one tool in particular, the PFORT verifier, has been widely used to insure portability. Tools for transforming programs to meet certain requirements have also been used in the production of numerical software; for example, in the production of LINPACK.

Program testing for performance evaluation has relied primarily on the use of selected test problems. Collections of test problems for several subject areas--ordinary differential equations, linear equations, and optimization--exist, and have received extensive use. Performance profiles have been used to present results of performance testing in a succinct and meaningful way. Modelling has been used to evaluate the performance of an algorithmic idea without incurring the expense of actually executing all of the supporting software.

THE APPLICATION OF HALSTEAD'S SOFTWARE SCIENCE DIFFICULTY MEASURE TO A SET OF PROGRAMMING PROJECTS

by

Charles P. Smith
International Business Machines Corporation
General Products Division
Santa Teresa Laboratory
San Jose, California

The difficulty measure, as defined by Halstead, is shown to have useful applications in the measurement of certain properties in code implementations. This difficulty measure reflects the effort required to code an algorithm, to test it, to inspect and review it, and to understand it later when the code needs alterations.

This paper explains how the difficulty metric reveals insights to the structure of a program module and also to some possible code impurities within the module. It also shows that assembler language programs are significantly more difficult than higher-level PL/S language programs. The author proposes that a maximum level (or threshold) of difficulty should be established to help manage the complexity of programs.

MATHEMATICAL PROGRAMMING ALGORITHM IN APL

by

Harlan Crowder
IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598

The APL programming language offers a way of succinctly expressing data processing algorithms. We will introduce and demonstrate those APL concepts which we find useful for designing, implementing, and testing mathematical programming procedures.

least-squares problems. This is unfortunate, since it would be desirable to choose tuning constants by tests on a large number of "typical" problems. Unable to do so, I generally resort to using a large collection of test problems from the literature, usually with the variations on starting guesses recommended in [MorGH81]. (While working on NL2SOL, I used the test problems listed in [GilM78], and while working on [Gay80], I used the unconstrained minimization test problems described in [MorGH81].)

Most of the time when I compare two algorithmic variations, neither performs uniformly better than the other. In fact, I now suspect that if one conducts a series of comparisons and finds that one algorithm performs uniformly better than the others, then this may mean that the set of test problems is too small.

Sometimes it seems appropriate to generate "random" test problems. I did this, for example, when preparing [Gay81]. When conducting tests that involve a random-number generator and that may later be written up for the benefit of other researchers, I strongly believe that one should use a portable random-number generator; and in writing up the results of such testing, one should state precisely how the test problems were generated (specifying, in particular, the random number generator and seed used). This way other researchers, who may be using other kinds of computers, can duplicate the test problems (except, perhaps, for roundoff error).

strategies, some of which are described in [DenGW80], and we also conducted limited experiments with a few tuning constants.

It should be obvious that one should vary only one detail at a time, since otherwise it may be hard to tell what effect individual details have on performance. Things would be simpler if all details were mutually orthogonal, so that by making a single test run (or set of test runs) for each detail, one could determine the best combination of all details. Unfortunately, the details may interact with one another, so that determining the best combination may require trying all combinations. Because of limitations on my time and budget, I do not try all combinations, but rather pick what I think are reasonable choices for all the details, then change one detail at a time from its "nominal" value. If I later decide to change my choice of nominal value for some detail, I may repeat varying some of the other details.

2. Choosing a set of test problems

It is desirable to see both how an algorithm behaves on pathological problems and how it behaves on "typical" problems. A number of the test problems that are mentioned repeatedly in the literature fall into the former category; the latter are much more difficult to identify. I know, for instance, of no simple way to quantify the space of "real life" unconstrained minimization problems or nonlinear

A STUDY OF REDUNDANCY IN MATHEMATICAL PROGRAMMING

by

Mark Karwan
Vahid Lotfi
State University of New York at Buffalo

Jan Telgen
Center for Operations Research and Econometrics

Stanley Zionts
State University of New York at Buffalo

This paper surveys methods for identifying and removing redundancy in mathematical programming and discusses a current project where empirical tests are performed on a number of the methods developed during the past twenty years. This project includes the development of computer programs and the implementation and testing of many prominent models. The complete results will be presented in a volume referenced in the paper.

SOLUTION STRATEGIES AND ALGORITHM BEHAVIOR IN LARGE-SCALE NETWORK CODES

by

Richard S. Barr
Edwin L. Cox School of Business
Southern Methodist University
Dallas, Texas 75275

This paper surveys results of a recent experiment testing different solution strategy parameters and implementation schemes of the U. S. Department of Treasury in/out-of-core, primal-simplex-based solution system for optimally merging micro-data files. Numerous runs were implemented to study pricing and pivoting rules, data storage technique, compiler, data page size, problem density, and algorithm behavior. Improvements over previous performance levels are noted.

RECURSIVE PIECEWISE-LINEAR APPROXIMATION METHODS FOR NONLINEAR NETWORKS

by

R. R. Meyer
Computer Sciences Department
University of Wisconsin,
Madison, WI

The use of recursive piecewise-linear approximations in network optimization problems with convex separable objectives allows the utilization of fast solution methods for the corresponding linear network subproblems. Piecewise-linear approximations, including the ability to utilize simultaneously information from infeasible as well as feasible points (so that results of Lagrangian relaxations may be directly employed in the approximation), guaranteed decrease in the objective without the need for any line search, and easily computed and tight bounds on the optimal value. The speed of this approach will be exemplified by the presentation of computation results for large-scale nonlinear networks arising from econometric and water supply system applications.

COMPUTATIONAL TESTING OF ASSIGNMENT ALGORITHMS

by

Michael Engquist
Department of Mathematics & Computer Science
Eastern Washington University
Patterson Hall
Cheney, WA 99004

The methods used in an empirical evaluation of several recently proposed assignment algorithms are discussed. Brief descriptions of the algorithms tested are also included.

am interested in studying how various techniques for solving minimization problems and nonlinear equations compare with one another. Some of the papers I read compare computer codes implementing different algorithms; such comparisons are complicated by the fact that the codes compared usually differ in several respects, and it may be hard to decide how much weight to give to the various differences. The testing that I conduct, by contrast, usually involves changing just one algorithmic detail at a time. Though this kind of testing is simpler than comparing totally different computer codes, it is still nontrivial, and it will be the subject of the rest of this paper.

There are several aspects to the kind of computational testing with which I am concerned. These include:

1. Deciding what to vary.
2. Choosing a set of test problems.
3. Choosing performance indicators.
4. Running the tests.
5. Analyzing the results.

Let us examine each of these in more detail.

1. Deciding what to vary

The algorithms that I study generally include a number of details that it may be interesting to vary. These include both tuning constants and algorithmic strategies. In the course of the work that went into NL2SOL [DenGW80], for example, we had occasion to vary a number of algorithmic

COMPARING ALGORITHMIC VARIATIONS

David M. Gay*

Abstract

While working on the nonlinear least-squares solver NL2SOL and other codes for unconstrained minimization and systems of nonlinear equations, the author has had occasion to compare the effectiveness of various alternative algorithmic details. This paper describes the test procedures and method of analysis used and expresses opinions on various relevant topics.

Introduction

This paper was written in response to an invitation to speak on computational testing at an ORSA/TIMS meeting. It describes an approach to comparing algorithmic variations that I have used in several contexts and expresses some relevant opinions.

Much has been written lately about comparative testing of computer algorithms. For example, a number of good papers on this subject appear in [Fos79], the COAL Newsletter is largely concerned with various aspects thereof, and <u>ACM Transactions on Mathematical Software</u> often publishes papers on testing. Thanks to such work as [CroDM79] and [MorGH81], the techniques people use to carry out and report on software testing are, I think, improving.

I often read papers on computational testing, because I

----

*Center for Computational Research in Economics and Management Science, Massachusetts Institute of Technology, Cambridge, MA 02139. This work was supported by the National Science Foundation under Grant MCS79-06671.

"ESTABLISHING A GROUP FOR TESTING MATHEMATICAL PROGRAMS" A PANEL DISCUSSION

by

<u>CHAIRMAN</u>
John M. Mulvey
Princeton University

<u>PANELISTS</u>
Ron Dembo, Yale University
Richard Jackson, National Bureau of Standards
Darwin Klingman, University of Texas, Austin
Saul Gass, University of Maryland
Roy Marsten, University of Arizona
Ken Ragsdell, Purdue University
Ronald Rardin, Georgia Institute of Technology

Provided here is a summary of the comments made during this panel discussion. While every attempt was made to insure accuracy, J. Mulvey, who functioned as chairman during this discussion, claims full responsibility for any inaccuracies or misrepresentations which may persist. Names and addresses of all contributions to the discussion appear at the end of this text.

A MODEL FOR THE PERFORMANCE EVALUATION IN COMPARATIVE STUDIES

by

Klaus Schittkowski
Institute for Angewandte Mathematik und Statistik
Universitut Wurzburg
8702 Wurzburg WEST GERMANY

The paper describes a model for the performance evaluation in an arbitrary comparative study of mathematical algorithms. Distinctions will be made between "static" testing, i.e. evaluation of the design of the corresponding programs and "dynamic" testing i.e. evaluation of the performance of these programs when implemented and run on a digital computer. For both levels, relevant criteria for choosing among codes are described. Advantages of real life, hand selected and randomly generated test problems, respectively are outlined. It is furthermore, shown how a decision maker can combine a variety of performance measures into a final score for each major criterium and program.

SYSTEMATIC APPROACH FOR COMPARING THE COMPUTATIONAL SPEED OF UNCONSTRAINED MINIMIZATION ALGORITHMS

by

S. Gonzalez

The comparison of mathematical programming algorithms is inherently difficult, so that it is not easy to derive general conclusions about the relative usefulness, applicability, and efficiency of different algorithms. The problem is complicated by the variety of approaches used to compare algorithms. Most often, some of the approaches ignore some essential aspects of the comparison or fail to provide sufficient information about the following items: (a) clarification of the objectives of the comparison; (b) clear and complete description of the algorithms being compared; (c) specification of the memory requirements; (d) use of the same experimental conditions for all of the algorithms being compared; (e) sufficient information about the experimental conditions and the numerical results, so as to make them easily reproducible; (f) use of enough performance indices to ensure the fulfillment of the objectives of the comparison; (g) use of a reasonably large set of test problems having different characteristics; (h) clarification of the way in which the derivatives are computed; (i) measurement of the computational speed; (j) measurement of the effect of the stopping conditions; (k) measurement of the sensitivity to scaling; (l) presentation of the convergence rates; (m) use of several nominal points for each test problem; and (n) use of a standard format to present the result of the comparison.

Concerning the measurement of the computational speed, the use of the time-equivalent number of function evaluation N_e is recommended. A method of comparing the computational speed of unconstrained minimization algorithms has been developed, that uses N_e as a performance index. A step-by-step description of the method is given below.

[12] K. Schittkowski, "Nonlinear programming codes: information, tests, performance", Lecture Notes in Economics and Mathematical Systems 183, Springer-Verlag, New York, New York, 1980.

[13] N. Z. Shor, "New development trends in nondifferentiable optimization", (in Russian), Kibernetika 13 1977 87-91, translated in Cybernetics 13 1977 881-886.

[14] N. Z. Shor, "Cut-off method with space extension in convex programming problems", (in Russian) Kibernetika 13 1977 94-95 translated in Cybernetics 13 1977 94-96.

[15] N. Z. Shor and V. I. Gershovich, "Family of algorithms for solving convex programming problems", (in Russian), Kibernetika 15 1979 62-67, translated in Cybernetics 15 1980 502-508.

[16] R. D. Wiebking, "Deterministic and stochastic geometric programming models for optimal engineering design problems in electric power generation and computer solutions", Technical Report TR73LS132, General Electric Company, Schenectady, New York, 1974.

Acknowledgement

Research supported in part by the National Science Foundation, Grant No. MCS78-02096.

References

[1] P. A. Beck and J. G. Ecker, "Some computational experience with a modified convex simplex algorithm for geometric programming", Report ADTC-72-20, Armament Development and Test Center, USAF Systems Command, Eglin AFB, Florida, 1972.

[2] R. S. Dembo, "A set of geometric programming test problems and their solutions", Mathematical Programming 10 1976 192-213.

[3] E. D. Eason and R. G. Fenton, "Testing and evaluation of numerical methods for design optimization", UTME-TP 7204, University of Toronto, 1972; also see E. Eason, "Evidence of fundamental difficulties in nonlinear optimization code comparisons", Failure Analysis Associates, Palo Alto, California, 1981.

[4] J. G. Ecker, "A geometric programming model for optimal allocation of stream dissolved oxygen", Management Science 21 1975 658-668.

[5] J. G. Ecker and M. Kupferschmid, "An ellipsoid algorithm for convex programming", 1981, presented at CORS-TIMS-ORSA Joint National Meeting, Toronto, 5 May 1981 (Session TD26.3).

[6] J. G. Ecker and R. D. Niemi, "A dual method for quadratic programs with quadratic constraints", SIAM Journal of Applied Mathematics 28 1975 568-576.

[7] L. Fox and J. H. Wilkinson, "NAG Fortran library manual", Mark7, Numerical Algorithms Group Ltd., Oxford, England, 1978.

[8] D. W. Hearn, private communication to J. G. Ecker, 1976.

[9] D. M. Himmelblau, "Applied nonlinear programming", McGraw-Hill, New York, New York, 1972.

[10] L. G. Khachian, "A polynomial algorithm in linear programming", (in Russian) Doklady Akademiia Nauk SSSR 244 1979 1093-1096, translated in Soviet Mathematics Doklady 20 1979 191-194.

[11] B. T. Poljak, Institute for Control Science, Moscow, USSR, June 22, 1981, private communication.

THE EVALUATION OF OPTIMIZATION SOFTWARE FOR ENGINEERING DESIGN

by

K.M. Ragsdell, P. E.
Visiting Professor
Aerospace and Mechanical Engineering
The University of Arizona
Tucson, Arizona

The design and implementation of two major comparative experiments is reviewed with particular emphasis on testing methodology. These studies took place at Purdue University. The first, an investigation of the merits of general purpose nonlinear porgramming codes with major funding from the National Science Foundation, was conducted over the period 1973 to 1977. The second, an investigation of the relative merits of various geometric programming strategies and their code implementations with funding from the Office of Naval Research, was conducted over the period 1974 to 1978. The various major decisions associated with such studies are discussed; such as the selection and collection of problems and codes, the nature of data to be collected, evaluation criteria, ranking schemes, presentation and distribution of results, and the technical design of the experiment itself. The statistical implications of the results in light of the experiment design are examined; as are the effects of various experiment parameters such as number of variables, number of constraints, degree of nonlinearity in the objective and constraints, and starting point placement. Finally, recommendations for future experiments are given.

STANFORD UNIVERSITY
STANFORD, CALIFORNIA 94305

SYSTEMS OPTIMIZATION LABORATORY
DEPARTMENT OF OPERATIONS RESEARCH

April 24, 1981

Karla Hoffman
Center for Applied Mathematics
National Bureau of Standards
Washington, DC 21234

Dear Editor:

I read with interest the article by M.J.D. Powell and the response by R. Jackson. I am much in agreement with the sentiments expressed by Jackson. Software libraries such as NAG's are now distributed worldwide to hundreds of computing sites. Even given a high quality routine, it requires a considerable effort by library organizations to include new material (and to delete obsolete routines). I do not think that such libraries should be viewed as test beds for new unproved algorithms, no matter how promising to their discoverer. If the discoverer of an algorithm is not prepared to expend the effort required to carefully implement and test his algorithm, he can hardly expect others to behave differently. It would seem to me that anyone with a serious interest in software needs to develop his own contacts. He needs to be directly in touch with real applications of optimization. Even given the willingness of libraries to include material that has not been thoroughly tested, the inevitable delays in distribution make any response from such users of marginal value.

Sincerely,

Walter

Walter Murray

WM/yp

cc: R. Jackson

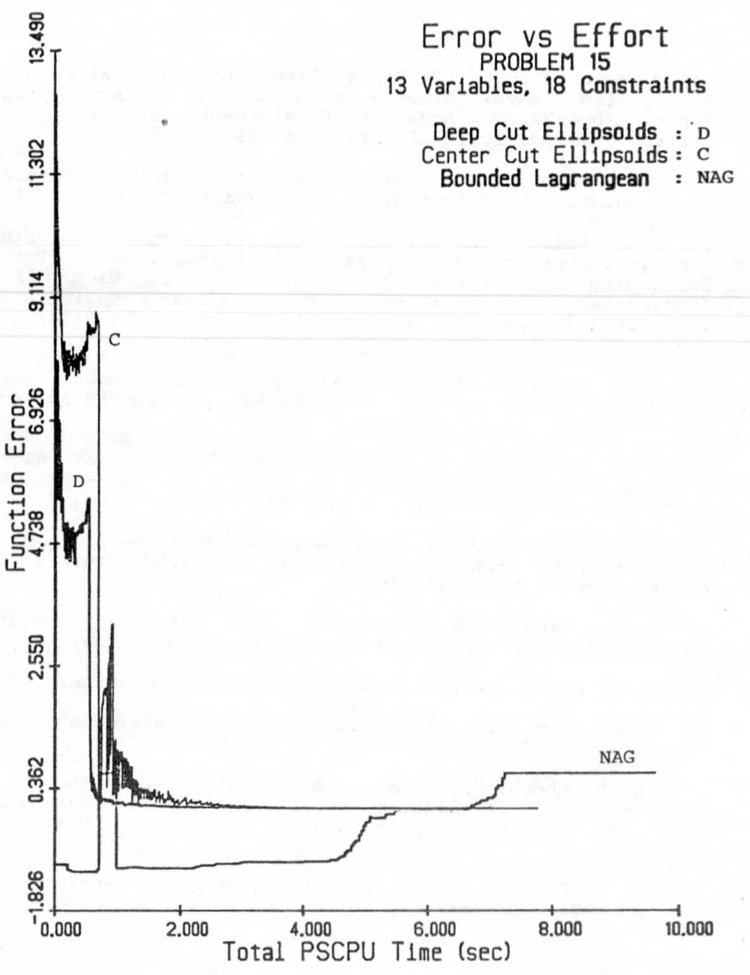


FIGURE 2

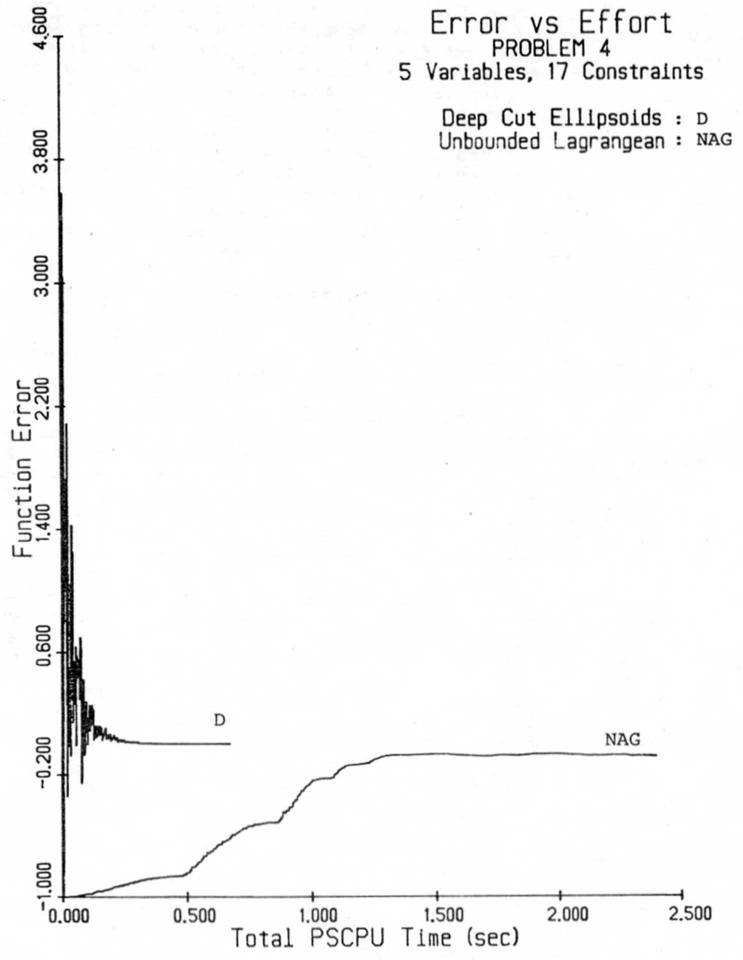


FIGURE 1

March 2Ø, 1981

To the Editor

I would like to establish the following litmus test for future membership in COAL:

Question 1. You are sent a paper to review from an editor of a prestigious journal on optimisation. It is written in Chaucerian English and Roman numerals are used in place of Arabic numerals. The paper seems to present advances to the state of the art. Which of the following would you do?

a. Send it back requesting that it be written in a dialect more familiar to the current readership.

b. Find a Chaucerian scholar and have it translated.

c. Find it wonderful that such diversity exists among the scholars of optimisation.

d. Conclude that Geoffrey Chaucer had insights into optimisation and read Canterbury Tales.

e. None of the above.

Question 2. Your discipline is a science (e.g., physics, chemistry, biology, sociology,...). You are sent a paper to review from an editor of a pretigious journal. It contains the experiments used to support the author's latest theory. It states that the author chose several specimens (but not why) and played in the laboratory until he had produced results that were better than previously reported. Which of the following would you do?

a. Send the paper back to the author stating he should be more scientific.

b. Applaud the breakthrough and recommend that it be published immediately to advance science.

c. Nominate the author for a Nobel.

d. None of the above.

[illegible]
Richard P. O'Neill

University of Wisconsin-Madison

Department of Industrial Engineering
1513 University Avenue
Madison, Wisconsin 53706
Telephone: 608/262-2686

16 March 1981

Dr. Karla L. Hoffman
Center for Applied Mathematics
National Bureau of Standards
Washington, D.C. 20234

Dear Dr. Hoffman:

I am taking this opportunity to respond to your invitation in the February 1981 issue of the Committee on Algorithms Newsletter, for readers to submit their opinions about the publications standards controversy.

I believe that the paper by Powell and the comments by R.H.F. Jackson proceed from two very different ideas about the functions of journals in our profession. In fact, I think much of the disagreement can be explained (although not resolved) by reference to these two different viewpoints. Perhaps I can briefly summarize how I see each point of view, and then make some comments about them.

I think the first point of view is that journals are primarily for the quick communication of currently relevant information, and that their historical significance is definitely secondary to this aim. The people who adopt this point of view would naturally take the position that papers to be published in journals should be correct and significant, but that they need not be final in the sense that they may communicate partial information, or they may contain algorithms or other methods which have not been completely tested (and indeed which cannot be completely tested without being communicated to others in the profession). I understand this to be fairly close to Powell's position.

The second point of view, which I understand Jackson to be taking, would hold that journals exist primarily to provide a historical record of research performed. To persons of this view, "standards" should guard against the publication of incompletely developed work, and the timeliness of publication of a paper is much less important than its degree of polish and perfection.

Although the above statements are obviously and deliberately oversimplified, I think they represent points of view that are essentially different and that adequately account for the existence of the kind of controversy reported in the February issue. One would expect people holding these views to differ strongly about the publishability of precisely the kinds of work that Powell discusses in his paper. Therefore, I think it is more profitable to interpret controversies like these as differences over the fundamental purposes of journals than as differences about particular issues such as software.

For each of the test problems, [5] gives an error versus effort curve comparing the two algorithms for efficiency. Two of these curves, for problems 4 and 15, are reproduced in Figures 1 and 2. From inspection of these curves it is clear that the EA finds solutions of modest accuracy much more quickly than NAG, and this behavior was displayed on all of the other problems as well. Figure 2 contains error curves for two variations of the EA, namely when center cuts are used for feasibility steps and when deep cuts are used. These curves show that for this problem deep cuts result in slightly faster EA convergence, but in many instances it is found that deep cuts are not worthwhile because the saving in iterations is more than offset by the additional work required for the line search in each feasibility step.

On the basis of the experiments reported in [5], we conclude that the algorithm is robust and efficient for nonlinear programming problems. Although the largest problem attempted in the study contains 36 variables and 33 constraints, there is nothing in our observations to suggest that the algorithm would not be successful on much larger problems as well. This conclusion is in dramatic contrast to the consensus about ellipsoid algorithms which has regretably grown up within the mathematical programming community because of the failure of these algorithms to compete with the simplex method for practical linear programming problems. L. G. Khachian's use, [10], of an ellipsoid algorithm as a theoretical device in proving the existence of a polynomial time algorithm for linear programming was of great importance to the theory of computational complexity, and because Khachian's paper was the first introduction to ellipsoid algorithms for most Western researchers, early computational studies naturally focused on linear programming. The time has come to give more attention to the use of ellipsoid algorithms for solving nonlinear programming problems, which was the original intent of Shor (see [13], [14], and [15]). According to B. T. Poljak, [11], some computational experiments with ellipsoid algorithms for nonlinear programming have been attempted in the USSR, but to the best of our knowledge no other researchers in the West have done such experiments.

The test problems consist of 8 quadratically constrained quadratic programming problems and 7 geometric programming problems, as summarized in the following table:

Problem	# variables	# constraints	reference
1	10	2	
2	5	4	[6]
3	3	3	[8]
4	5	17	[9], #11, p406
5	15	3	
6	10	1	
7	10	1	
8	15	1	
9	5	2	[1], #6, p29
10	7	3	[1], #12, p34
11	7	7	[1], #13, p35
12	24	42	[1], #16, p41
13	15	13	[4]
14	36	33	[16], #5, p217
15	13	18	[2], #6, p206

Problems 4 and 15 are nonconvex.

For each of the 15 test problems, the ellipsoid algorithm obtains a strictly feasible solution point with an objective function value near its optimum, and good feasible points are usually found early in the EA iterations. This robust performance contrasts with that displayed by NAG. Even though NAG requires only epsilon feasibility, and sometimes fails to find any strictly feasible points at all, it frequently converges to points having higher objective values than those obtained by the EA. On problems 10 and 15, for example, NAG's outputs stop changing at points which have objective values 5% and 55% higher, respectively, than those obtained by the EA. On problem 14, NAG converges to a grossly infeasible point, with most of the variables at their upper or lower bounds. No attempt was made to adjust NAG's parameters so as to obtain better performance, but the fact that NAG has difficulty with a significant fraction of the test problems when the suggested parameter values are used indicates that the problems are hard enough to be a fair test of both algorithms. For some of the problems NAG was permitted to enforce bounds on the variables, but this did not often result in better performance; for example, in problem 12 of [5], enforcing bounds leads to an addressing exception within a NAG subsidiary routine. It is notable that while EA convergence cannot be guaranteed for nonconvex problems, the algorithm is frequently successful on such problems.

For what it may be worth, I think that the first position is much preferable to the second. Personally, I do not think that very many people fifty years from now will be reading the journal articles that we publish today, or at least not most of them. How many people now read the work in mathematics that was published fifty years ago? Very few, because the part of that work that is of permanent relevance has mostly found its way into texts or research monographs, or in other ways has become part of our working knowledge. Therefore, I see the historical functions of journals as very definitely secondary to their crucial function of quickly and widely disseminating information. It follows that I think it is perfectly appropriate, and indeed very important, for journals to publish incompletely developed work, partially tested codes, or anything else that will help to disseminate needed information. I believe that those who set up so-called "standards" for the purpose of preventing this kind of publication do all of us a great disservice.

Of course, this is not an argument against all standards. We need to make sure that work published in journals does not contain gross errors, and that it is significant enough to warrant the expenditure of money to publish and disseminate it. These functions are served by the refereeing process, which is as applicable to codes and software as to mathematics. But the creation and enforcement of "standards" which aim at the publication only of polished contributions to the historical record is a great mistake. Not only does it result in a largely useless product, but it presents journals from performing their truly useful function of distributing, in real time, the results of current research.

Sincerely,

[illegible]

Stephen M. Robinson
Professor

SMR:jt

Calendar of mathematical programming meetings
as of 11 September 1981

Maintained by the Mathematical Programming Society (MPS)

This Calendar lists meetings specializing in mathematical programming or one of its subfields in the general area of optimization and applications, whether or not the Society is involved in the meeting. (These meetings are not necessarily "open".) Any one knowing of a forthcoming meeting not listed here is urged to inform the Vice Chairman of the Society, Dr. Philip Wolfe, IBM Research 33-221, POB 218, Yorktown Heights, NY 10598, U.S.A; Telephone 914-945-1642, Telex 137456.

Some of these meetings are sponsored by the Society as part of its world-wide support of activity in mathematical programming. Under certain guidelines the Society can offer publicity, mailing lists and labels, and the loan of money to the organizers of a qualified meeting. For further information address the Chairman of the Executive Committee, Dr. A. C. Williams, Computer Science Department, Mobil Oil Co. Technical Center, Box 1025, Princeton, New Jersey 08540, U.S.A.; Telephone 212-883-7678.

Substantial portions of regular meetings of other societies such as SIAM, TIMS, and the many national OR societies are devoted to mathematical programming, and their schedules should be consulted.

1981

October 19-20: Second Mathematical Programming Symposium Japan, Kyoto, Japan. Contact: Professor Toshihide Ibaraki, Department of Applied Mathematics and Physics, Faculty of Engineering, Kyoto University, Sakyo-ku, Kyoto, Japan 606.

October 19-22: "International Symposium on Optimum Structural Design" (Eleventh Naval Structural Mechanics Symposium), Tucson, Arizona, U.S.A. Contact: Dr. Erdal Atrek, Dept. of Civil Engineering, Building 72, University of Arizona, Tucson, AZ 85721, U.S.A.

1982

May 13-14: "Optimization Days", Campus of the University of Montreal. Contact: Professor Jacques Ferland, Centre de recherche sur les transports, Université de Montréal, C.P. 6128, Succ. "A", Montréal, Québec, Canada H3C 3J7; telephone 514-343-7575. Deadline for abstract, January 31, 1982. Sponsored by IEEE, ACFAS, SIAM, SCMA, and the MPS.

August 23-28: Eleventh International Symposium on Mathematical Programming in Bonn, Federal Republic of Germany. Contact: Institut für Ökonometrie und Operations Research Universität Bonn, Nassestraße 2, 5300 Bonn 1, Federal Republic of Germany; Telex 886657 unibo b, Telephone (02221) 739285. Official triennial meeting of the MPS. (Note: The International Congress of Mathematicians will be held August 11-19 in Warsaw, Poland.)

Successful Solution of Nonlinear Programs by an Ellipsoid Algorithm

by

J. G. Ecker and M. Kupferschmid

Mathematical Sciences Department
Rensselaer Polytechnic Institute
Troy, New York 12181

In [5], we investigate a variant of the ellipsoid algorithm for convex programming, discuss its computer implementation and the measurement of its performance, and present some computational results to show that it is robust and efficient compared to a widely used commercial code. Here we summarize those computational results.

The algorithm described in [5] is a variant of that introduced by Shor, [14]. For an optimization step, the cutting hyperplane passes through the center of the current ellipsoid, and this is called a center cut. Feasibility cuts can either be center cuts or the cutting hyperplane can pass between the current center and the feasible region, its closeness to the feasible region depending on the precision of a line search. When the cutting hyperplane separates the current center from the feasible region, the cut is called a deep cut.

Performance is evaluated by plotting curves of solution error versus computational effort, after Eason, [3]. Solution error is measured by the relative difference between the current objective function value and the (known) optimal value, and computational effort is measured by problem state central processor (pscpu) time used by the algorithm. A general method is elaborated in [5] for the measurement of cpu time, and for the segregation of that which is used by the algorithm from that which is used by non-algorithmic code. Experiments are reported which show that this method is effective in excluding from the measurements those contaminants most commonly cited as obstacles to the use of cpu time as a measure of effort.

The ellipsoid algorithm is compared to the augmented Lagrangean method implemented by subroutine E04VAF of the NAG subroutine library, [7], which is according to its documentation essentially identical to the routine SALQDR reviewed by Schittkowski, [12]. This commercial code is referred to below as "NAG", and the ellipsoid algorithm as the "EA". The parameters used for NAG are those recommended in its documentation; in particular, the experiments use the linesearch routine E04VAZ and the parameter "eta" is set to .9.

Other Future Technical Conference Sessions

- Jan Telgen is chairing a COAL-sponsored session at the EURO V/TIM XXV meeting in Lausanne, Switzerland, July 12-14, 1982.
- Al Williams is chairing a session at the Houston meeting of ORSA on Mathematical Programming Software Used by the Oil Industry.
- Earnest Eason is chairing a session at the Detroit meeting of ORSA/TIMS on Testing Methodologies.
- Plans are underway for a session at the San Diego meeting of ORSA/TIMS.

Testing Centers

- Ken Ragsdell's summary proposal for establishing such a center was discussed.
- Those at the meeting agreed that while COAL certainly supports such a concept in the same vein that we support any testing that conforms to our published guidelines, it would be inappropriate for COAL to become officially involved in establishing a testing center.

Computational Testing Prize

- It was agreed that a prize and award honoring exceptional computation testing efforts should be established. It would be awarded triennially with the first one to be given at the Bonn Symposium.
- Money for the first one will come from the leftover budget from the Boulder COAL Conference.

New Officers

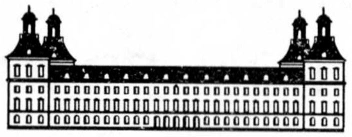
- Karla Hoffman was nominated as the new chairman of COAL.
- Jan Telgen was nominated as the new newsletter editor.
- Barring other nominations and pending approval of the MPS Executive Council, they will take office at the Bonn Symposium.

As before, I encourage all Friends of COAL (which you are if you receive this newsletter) to participate in any of out planned activities. If any of our readers would like to become involved in activities mentioned above, be it by presenting a paper in one of the planned sessions or assisting in collecting information about test problems and codes, please contact me or any other committee member. We actively seek the involvement of the mathematical programming community.

Richard H. F. Jackson

XI. INTERNATIONAL SYMPOSIUM ON MATHEMATICAL PROGRAMMING

UNIVERSITÄT BONN



AUGUST 23–27, 1982

Pre-Registration and First Call for Papers

The International Symposium on Mathematical Programming is the triennial scientific meeting of the Mathematical Programming Society. The XI. Symposium will be held at the University of Bonn, August 23–27, 1982. It will be organized by the Institut für Ökonometrie und Operations Research and the Sonderforschungsbereich 21 (DFG) of the Rheinische Friedrich-Wilhelms-Universität Bonn.

International Program Committee

B. Korte (W. Germany) Chairman
J. Abadie (France)
A. Auslender (France)
M. Avriel (Israel)
E. Balas (USA)
M. L. Balinski (France)
E. M. L. Beale (England)
J. F. Benders (Netherlands)
B. Bereanu (Romania)
C. Berge (France)
R. Burkard (Austria)
A. Charnes (USA)
V. Chvátal (Canada)
R. W. Cottle (USA)
G. B. Dantzig (USA)
M. A. H. Dempster (England)
L. C. W. Dixon (England)
S. Erlander (Sweden)
R. Fletcher (Scotland)
A. Geoffrion (USA)
F. Giannessi (Italy)
J.-L. Goffin (Canada)
E. Golstein (USSR)
R. L. Graham (USA)
P. L. Hammer (Canada)
M. Held (USA)
P. Huard (France)
T. Ibaraki (Japan)
M. Iri (Japan)
R. G. Jeroslow (USA)
E. L. Johnson (USA)
P. Kall (Switzerland)
L. Kantorovich (USSR)
R. M. Karp (USA)
V. Klee (USA)
A. Korbut (USSR)
K. O. Kortanek (USA)
J. Krarup (Denmark)
H. W. Kuhn (USA)
A. Land (England)
E. L. Lawler (USA)
C. Lemarechal (France)
F. Lootsma (Netherlands)
L. Lovász (Hungary)
G. P. McCormick (USA)
O. L. Mangasarian (USA)
B. Mond (Australia)
G. L. Nemhauser (USA)
F. Nozicka (CSSR)
W. Oettli (W. Germany)
A. Orden (USA)
M. W. Padberg (USA)
B. T. Poljak (USSR)
M. J. D. Powell (England)
A. Prékopa (Hungary)
W. R. Pulleyblank (Canada)
M. R. Rao (India)
K. Ritter (W. Germany)
S. M. Robinson (USA)
R. T. Rockafellar (USA)
J. B. Rosen (USA)
H. E. Scarf (USA)
J. Stoer (W. Germany)
A. W. Tucker (USA)
S. Walukiewicz (Poland)
R. Wets (USA)
A. Wierzbicki (Poland)
P. Wolfe (USA)
L. Wolsey (Belgium)

ex officio
A. Bachem (W. Germany)
M. Grötschel (W. Germany)
Co-Chairmen of the Organizing Committee

(Acceptance of invitation received by July 1, 1981)

Call for papers:
Papers on all theoretical, computational and applied aspects of mathematical programming are welcome. The presentation of very recent results is encouraged. For this reason a particulary late deadline for the submission of abstracts has been set.

Dates and Deadlines:
We kindly ask you to return the pre-registration form at your earliest convenience, but not later than October 1, 1981. This will be a great help for the organizers.
Deadline for contributed papers: April 1, 1982, the abstracts of the papers are due by June 1, 1982.

Topics:
Sessions on the following topics will be organized. Suggestions for further areas to be included in the program are welcome.
— Linear, Integer, Mixed Integer Programming
— Nonlinear, Nonconvex, Nondifferentiable Optimization
— Combinatorial Optimization, Networks, Graph Theory
— Dynamic Programming, Stochastic Programming, Optimal Control Theory
— Complementarity and Fixed Point Theory
— Game Theory and Multicriterion Optimization
— Computational Complexity, Approximative Methods, Heuristics
— Implementation and Evaluation of Algorithms and Software
— Teaching of Mathematical Programming
— Applications of Mathematical Programming in Economics, Management, Engineering, Industry, Government, Traffic and Transportation, Natural and Human Sciences, Energy and Agriculture

Mailing Address:
Math. Progr. Secretariat
c/o Institut für Operations Research
Nassestrasse 2
D-5300 Bonn 1
W. Germany
Telefon: (0228) 739203, Telex: 886657 unibo d

Structure of the Meeting:
The meeting will offer contributed as well as invited papers. In addition, state-of-the-art tutorials are planned. These will be lectures of one hour's duration on subjects of mathematical programming which have developed rapidly in recent years and for which there is little information for the nonspecialist. These tutorials are intended to introduce the nonspecialist to the subject and to guide him toward the most interesting results and areas of further research.
The Dantzig Prize and the Fulkerson Prize will be awarded during the symposium.

Site:
The symposium will take place in the main building of the University of Bonn, a former residential palace. Bonn is attractively located on the river Rhine and is easily accessible by air, train or car.

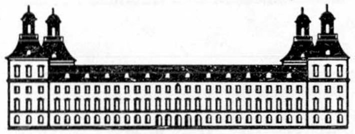
Further Announcements:
Further information regarding the symposium will be sent in due course to all those who have pre-registered. This will include additional information about the program, registration fees, social events, travel and hotel information, etc.

Achim Bachem
Martin Grötschel
Co-Chairmen
Organizing Committee

Bernhard Korte
Chairman
Program Committee

XI. INTERNATIONAL SYMPOSIUM ON MATHEMATICAL PROGRAMMING

UNIVERSITÄT BONN



AUGUST 23–27, 1982

Pre-Registration and First Call for Papers

Please return this form to: **Math. Progr. Secretariat**
c/o Institut für Operations Research
Nassestrasse 2
D-5300 Bonn 1
W. Germany

PRE-REGISTRATION FORM

Please keep me on the mailing list for further information about the Eleventh International Symposium on Mathematical Programming.

Last Name: ______________________ I plan to attend the symposium ○ yes ○ no

First Name: ______________________

Institution: ______________________ I intend to give a talk ○ yes ○ no

Street: ______________________ My talk will be on the following subject ______________________

City: ______________________ ______________________

Country: ______________________ ______________________

CHAIRMAN'S COMMENTS

The last few months have been spent primarily in planning. A COAL business meeting was held at the National Bureau of Standards on August 17. Topics discussed ranged from plans for the next three years to nominations for chairman and newsletter editor. The minutes of that meeting have been sent to all committee members and are available on request of that meeting have been sent to all committee members and are available on request to all other interested parties. Salient points of that meeting are summarized below.

Future Directions

- Continue to collect information on test problem availability and whereabouts
- Do the same for software
- Publish these either in the newsletter or in a separate catalogue
- Develop standardized test problem formats
- Develop classification schemes for test problems

Next COAL Conference

- Jan Telgen, Klaus Schittkowski, and Susan Powell are working on organizing the next conference on Testing MP Software to follow the first and second ones held respectively in Sogesta, Italy in 1977 and in Boulder, CO in January 1981.
- The current plan is to try for NATO funding for an Advanced Research Institute in 1983 or 1984.

Technical Sessions at MPS Symposium in Bonn, August, 1982

- Pending approvel of the Program Committee, we plan the following sessions:

  Recent results in software testing I
  Recent results in software testing II
  Tools for software testing
  State-of-the-art in testing methodologies
  Standardized test problem formats and classification schemes
  Open business meeting

COMMITTEE ON ALGORITHMS of the
MATHEMATICAL PROGRAMMING SOCIETY

CHAIRMAN

Richard H. F. Jackson
Center for Applied Mathematics
National Bureau of Standards
Washington, D. C. 20234
(301) 921-3855

EDITOR OF THE NEWSLETTER

Karla L. Hoffman
Center for Applied Mathematics
National Bureau of Standards
Washington, D. C. 20234
(301) 921-3855

Jacques C. P. Bus
Haagwinde 61
1391 XX
Abcoude, The Netherlands

Harlan P. Crowder
IBM Thomas J. Watson Research Center
P. O. Box 218
Yorktown Heights, NY 10598
(914) 945-1710

Jan L. de Jung
Amerikalaan 83
5691 KC Son
Nederland

Leon S. Lasdon
Department of General Business
School of Business Administration
Austin, TX 78712
(512) 471-3322

John M. Mulvey
School of Engineering/Applied Science
Princeton University
Princeton, NJ 98540
(609) 452-5423

Richard P. O'Neill
EI 622, Room 4447
Department of Energy
Washington, DC 20461

Susan Powell
University of Kent at Canterbury
Rutherford College
Kent CT 2 7NX
England

Patsy B. Saunders
Center for Applied Mathematics
National Bureau of Standards
Washington, DC 20234
(301) 921-3855

Klaus Schittkowski
Institute fur Angewandte
Mathematik und Statistik
Universitat Wurzburg
D-87 Wurzburg
West Germany

Jan Telgen
RABOBANK NEDERLAND
Department of Applied Mathematics
Laan Van Eikenstein 9 (ZL-G-170)
3705 AR Zeist
The Netherlands

EX OFFICIO MEMBERS

J. Abadie, Chairman, MPS
29, Boulevard Edgar-Quintet
75014 Paris, France

A. C. Williams, MPS Exec. Comm.
Mobil Oil Co. Technical Center
P. O. Box 1025
Princeton, NJ 08540

Philip Wolfe, Vice-Chairman, MPS
IBM Research 33-2
P. O. Box 218
Yorktown Heights, NY 16598

COAL OBJECTIVES

The Committee on Algorithms is involved in computational developments in mathematical programming. There are three major goals: (1) ensuring a suitable basis for comparing algorithms, (2) acting as a focal point for computer programs that are available for general calculations and for test problems, and (3) encouraging those who distribute programs to meet certain standards of portability, testing, ease of use, and documentation.

NEWSLETTER OBJECTIVES

The newsletter's primary objective is to serve as a forum for the Friends of COAL. Through an informal exchange of opinions, members have an opportunity to share their experiences. To date, our profession has not developed a clear understanding on the issues of how computational tests should be carried out, how the results of these tests should be presented in the literature, or how mathematical programming algorithms should be properly evaluated and compared. These issues will be addressed in the newsletter.

DISTRIBUTION OF THIS NEWSLETTER

This newsletter is mailed to every member of the Mathematical Programming Society and to all "friends" of COAL. If you are not presently receiving this newsletter and would like to, please write to the editor requesting that your name be added to the list of "friends" of COAL. There is currently no charge for this newsletter.

Mathematical Programming Society
Committee on Algorithms Newsletter

No. 6: September 1981 Karla L. Hoffman, Editor

Contents:

DR. KARLA L. HOFFMAN
UNITED STATES DEPARTMENT OF COMMERCE
NATIONAL BUREAU OF STANDARDS
CENTER FOR APPLIED MATHEMATICS
WASHINGTON, D.C. 20234

DEC 7 1981
PM

R. M. THRALL
12003 PEBBLE HILL DRIVE
HOUSTON, TEXAS 77024
USA

PRIORITY MAIL

~~THIRD CLASS MAIL~~