RECOMMENDATIONS FOR THE DESIGN OF NONLINEAR PROGRAMMING CODES

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

The underlying mathematical model to be solved is the following type of nonlinear programming problem:

$$\min f(x)$$

$$x \in R^n: \quad \begin{aligned} g_j(x) &= 0 \quad , \quad j = 1, \ldots, m_e \\ g_j(x) &\geq 0 \quad , \quad j = m_e + 1, \ldots, m \end{aligned} \qquad \text{(NLP)}$$

The functions f and g_j, $j = 1, \ldots, m$, are assumed to be continuously differentiable.

A set of 26 NLP optimization programs was collected and implemented by the author to test their performance on a variety of test problems. This set of codes consists of realizations of penalty function methods, multiplier methods, generalized reduced gradient methods, quadratic approximation methods (solution of a quadratic subprogram in each iteration), and of Robinson's method (minimization of the Lagrangian subject to linearized constraints in each iteration). A more detailed description of their underlying mathematical structure and information about the technical organization of the codes are contained in [1]. Nine different performance criteria (e.g., efficiency, reliability, ease of use, etc.) are defined and are evaluated by 370 test runs based on randomly generated test problems with predetermined solutions, cf. [2,3,4]. In addition, each program has to solve about 120 classical test problems which are used in earlier comparative studies, cf. Hock and Schittkowski [5,6].

Based on extensive experience with the implementation and testing of optimization programs, I present the following recommendations for the design and organization of nonlinear programming codes. The views presented are based on a user outlook and discuss approaches for design of future software. I will not present any recommendations about the mathematical structure of the algorithm. I restrict myself only to some technical implementation details which could be realized in every program independently from the underlying algorithm. These recommendations apply to the following situation: a designer is developing a code with the intention of distributing it to non-specialized users and executing it on different computers.

Problem formulation: When looking for an adequate problem formulation, I recommend the use of the format (NLP) since most optimization programs within the comparative study proceed from this formulation. Nevertheless, any equivalent formulation could be used.

Programmer language: All programs which were submitted for the comparative study were written in FORTRAN. I therefore recommend that FORTRAN be used for any future programs, especially if the new code has to replace an existing one. Whenever possible, the designer should use a portable version of FORTRAN, for example ANSI FORTRAN. Furthermore, the program should not contain any auxiliary subroutines in machine language.

Precision: Since the machine precision varies from one computer to another, an optimization program should be available in single and double precision versions.

Documentation: Clearly written and detailed documentation facilitates the solution of problems for all users who are not familiar with the program. As a model, I recommend the documentation of the programs contained in the Optimization Software Library of the National Physical Laboratory. A documentation of this library consists of the following sections:

Purpose	Auxiliary routines
Description	Storage
Specification	Timing
Parameters	Accuracy
User-supplied subroutines	Further comments
Error indicators	Keywords
	References

In addition, implementation information and a detailed example are presented.

Provision of problem data: In general, the problem data such as the dimension of the problem, the number of constraints, tolerances, etc. could be provided either within a user-written driving program or read from an input deck. The second alternative (the input deck) is preferred if unsophisticated users have to solve individual problems. However, there could be some organization difficulties when the program is used to solve problems which are complicated. If a designer is doubtful, he should write the program in the form of a subroutine and supply the problem data within the main program in common areas or subroutine arguments. The best approach would be to offer both versions.

Provision of problem functions: The question of how an optimization program receives the problem functions depends on the underlying mathematical algorithm, but a designer should try to require the evaluation of as many problem functions and gradients as possible in one subroutine or block.

Numerical differentiation: It should be self-evident that a user-oriented optimization program possesses the capability of calculating derivatives numerically.

Linear constraints: The solution of a problem is facilitated if the user could supply linear constraints and bounds of the variables separately and not in the form of general equality or inequality restrictions. This feature reduces the probability of programming errors.

Parameters: An optimization algorithm requires the definition of parameters for stopping criteria, steplength procedures, maximum number of iterations, and so on. A designer should predetermine default values for each parameter which are dependent upon the underlying machine precision, and the user should be informed about possible alterations to these settings and their effects on the performance.

Flag: The flexibility of an optimization program is enhanced if the user has control over the entering and leaving of the subroutine calculating the problem functions. The implementation of a flag is therefore recommended.

Output: The output should consist of multiple print levels to follow the course of computations as closely as necessary. If the program terminates unsuccessfully (i.e. did not obtain a solution to the problem), the user should get complete error informations. On the other hand, the user should have the capability of suppressing all output.

Dimensioning: An optimization program should be variably dimensioned so that the required central memory is adapted to the problem size. As an example for variably dimensioned arrays, consider the FORTRAN subroutines in Lawson, Hanson [7].

[1] K. Schittkowski, Nonlinear programming codes - Information, test, performance -, submitted for publication.

[2] -, Randomly generated nonlinear programming test problems, to appear: Proceedings of the 9th IFIP Conference on Optimization Techniques, Warszawa, 1979.

[3] -, A numerical comparison of optimization programs using randomly generated test problems, in: Performance evaluation of numerical software, L. D. Fosdick ed., North-Holland Publishing Co., 1979.

[4] -, The performance of optimization programs for the solution of degenerate, ill-conditioned, and indefinite problems, submitted for publication.

[5] W. Hock, K. Schittkowski, Test examples for the solution of nonlinear programming problems. Part 1, Preprint No. 44, Institut fur Angewandte Mathematik und Statistik, Universitat Wurzburg, W. Germany, 1979.

[6] -, Test examples for the solution of nonlinear programming problems. Part 2, Preprint No. 45, Institut fur Angewandte Mathematik und Statistik, Universitat Wurzburg, W. Germany, 1979.

[7] C. L. Lawson, R. J. Hanson, Solving least squares problems, Prentice Hall, Englewood Cliffs, New Jersey, 1974.

****

WEIGHTS FOR THE EFFICIENCY AND THE ROBUSTNESS OF NONLINEAR OPTIMIZATION CODES

F. A. Lootsma, Department of Mathematics
University of Technology, 2600 AJ Delft, The Netherlands

In an earlier publication (Lootsma [1979]) we have demonstrated that Saaty's priority theory (Saaty [1977]), a well-known method for multicriteria decision analysis, can be successfully used to assess the performance of nonlinear optimization codes. First, the relevant performance criteria such as robustness, efficiency, capacity, ... are identified and weighted. Thereafter, the codes are compared and weighted under each of the performance criteria

- to obtain insight in the behavior of the methods and the effect of particular features,
- to select the "best" methods or combinations of methods for implementation as part of a software library.

To attain these ends, a methodology is developed for evaluating methods by testing the behavior of Newton-like programs (i.e. implementations on a certain computer of algorithms based on the methods). Part of the methodology included defining measures of performance such as "standard time" and quality of solution. The terms "robustness", "reliability", and "efficiency" are each defined and used within this study. Based on these notions, descriptions are given for the "representative test set" and "performance tables" provided for each Newton-like method. The actual evaluation leads to a qualitative discussion of robustness and reliability of the methods, which is based on results for representative test sets. A quantitative comparison of efficiencies is also presented. Special features such as conditional updating (use of quasi-Newton updates for the jacobian approximation in some iteration steps), scaling and reduction of problems with linear components are evaluated separately. Such features are, in fact, independent in the sense that they are applicable to a large class of Newton-like methods.

The final conclusions in [1] lead to a recommendation of two poly-algorithms for use in a software library. However, for particular problem classes one may prefer some other choice based on the comparative results presented in the paper. In fact, due to a methodologic set up, the test results have intrinsic value for the more general discussion of the use of Newton-like methods for solving systems of nonlinear equations.

[1] J. C. P. Bus, Numerical solution of systems of nonlinear equations, Mathematical Centre Tracts 122, Mathematisch Centrum, Amsterdam (1980).

A COMPARISON OF LEAST SQUARES AND LEAST ABSOLUTE DEVIATION REGRESSION MODELS FOR ESTIMATING WEIBULL PARAMETERS

K. D. Lawrence, ATT Long Lines, Bedminster, New Jersey
D. R. Shier, National Bureau of Standards, Washington, D. C.

This paper uses Monte Carlo simulation studies, embedded within an experimental design framework, to study alternative methods for simple estimation of paraemters of a Weibull distribution. In particular, least squares estimation (accomplished using a routine that forms a generalized universe) and least absolute deviation estimation (using a modification of the Barrodale and Roberts primal simplex code) were used to obtain Weibull estimates, based on a double-logarithmic transformation of the cumulative Weibull distribution function.

Simulated Weibull failure data were generated for a range of sample sizes and true parameter values. The parameter values were chosen to encompass a wide range of shapes for the Weibull distribution. Two slightly different linear models involving the two Weibull parameters were studied, as well as three different techniques for approximating the cumulative Weibull distribution.

For each data set so generated, estimates for the parameters were obtained using least squares and least absolute deviation regression techniques. For fixed sample size and selected parameter values, r = 100 repetitions of the Weibull life test were conducted. Based on these r repetitions, estimates were obtained for the bias and the mean squared error. Results of some 25,000 separate regressions indicate that L_1 estimation can provide more efficient estimates for Weibull failure data over the range of sample sizes and parameter values studied. Moreover, one of the linear models and two of the approximations for the cumulative distribution emerged as more suitable for this type of estimation.

****

DESIGN AND EVALUATION OF NEWTON-LIKE METHODS FOR THE NUMERICAL SOLUTION OF SYSTEMS OF NONLINEAR EQUATIONS

Jacques C. P. Bus, Mathematical Centre, P. O. Box 4079
1009 AB Amsterdam, The Netherlands

Newton-like methods are widely used for the solution of systems of nonlinear differentiable equations. In [1] an extensive theoretical and practical discussion of these methods is given. Starting from a formal definition of Newton-like methods, an extensive theory is developed concerning existence and uniqueness of solutions and global and local convergence properties. Based on this theory a large set of methods is designed and evaluated. The ultimate goal of the evaluation is:

separately. Multiplying the weights of the codes by the weights of the criteria, and adding the results, one obtains a final score for each code. The highest score designates the best code in the given circumstances.

In assigning weights to the codes under the performance criteria of efficiency and robustness, one will normally use the results of the comparative studies that have been published in the last 10 years. Nevertheless, the assignment is a non-trivial matter. Figure 1 shows a situation which is likely to occur as a result of computational experiments: a test battery of five codes and eight test problems, and such a dispersion of failures that the cross-section of test problems that have been successfully solved by all codes under consideration is empty. In figure 1 we have also displayed the minimum-time ratios introduced by Abadie and Guigou [1970]: taking t_{ik} to denote the execution time of code i on test problem k, then the minimum time ratio τ_{ik} is given by

$$\tau_{ik} = \frac{\min_i t_{ik}}{t_{ik}}$$

so that the highest score in a column designates the most efficient code for the particular test problem; moreover, these ratios provide some sort of scaling for test problems of increasing size.

In order to use the maximum amount of information in figure 1, we propose to consider each pair of codes separately. Let C(i,j) denote the cross-section of test problems that have been solved by the codes i and j. Now, we estimate the efficiency ratio of the codes i and j by the ratio of the row averages

$$r_{ij} = \sum_{k\in C(ij)} \tau_{ik} \Big/ \sum_{k\in C(i,j)} \tau_{jk} \quad .$$

To illustrate matters, we have C(1,2) = {3,6,7}, and

$$r_{12} = \frac{1.00 + 0.67 + 0.50}{0.60 + 0.50 + 0.42} = 1.43 \quad .$$

The resulting matrix of estimated efficiency ratios (the ratios of the row averages over the corresponding cross-sections) is displayed in figure 2, which also exhibits the results of the usual priority-theoretical analysis: the components of the Perron and Frobenius eigenvector (the weights of the codes under the performance criterion of efficiency), as well as the normalized row sums and the normalized inverted column sums to approximate these components. An interesting phenomenon appears in figure 3: the directed graph, with the nodes 1, . . . , 5 representing the codes, and the arcs representing the off-diagonal elements ≥ 1 in the matrix of estimated efficiency ratios, has so many cycles that the test set is obviously too small to draw definite conclusions about the efficiency of the codes. The cycle (1,5,4,3,1), for example, indicates that none of the codes 1,3,4,5 is consistently more efficient than the remaining ones.

The assignment of weights under the criterion of robustness could be straightforward. We normalize the percentages of successfully solved problems (63,75, 75,75,63) so that they sum up to unity; this leads to the weights 0.18, 0.21, 0.21, 0.21, and 0.18 respectively. In a forthcoming paper (Lootsma [1980]), however, we show that this procedure runs up against unexpected difficulties in test results with some success rates being slightly below and some being equal to 100 percent. The difficulty is that the weights have to express the preference of a decisionmaker for success rates below, say, 90 percent, between 90 percent and 100 percent, and equal to 100 percent.

References

[1] Abadie, J., and Guigou, J., Numerical Experiments with the GRG Method. Appendix III of J. Abadie (ed.), Integer and Nonlinear Programming. North-Holland, Amsterdam, 1970.

[2] Lootsma, F. A., Performance Evaluation of Non-Linear Programming Codes from the Viewpoint of a Decision Maker. In L. D. Fosdick (ed.), Performance Evaluation of Numerical Software. North-Holland, Amsterdam, pp. 285-297, 1979.

[3] Lootsma, F. A., Ranking of Non-linear Optimization Codes According to Efficiency and Robustness. To appear in L. Collatz, G. Meinardus, and W. Wetterling (eds.), Constructive Methods for Non-linear Optimization. Birkhauser, Basel, 1980.

[4] Saaty, Th. L., A Scaling Method for Priorities in Hierarchical Structures. J. Math. Psych. 15, pp. 234-281, 1977.

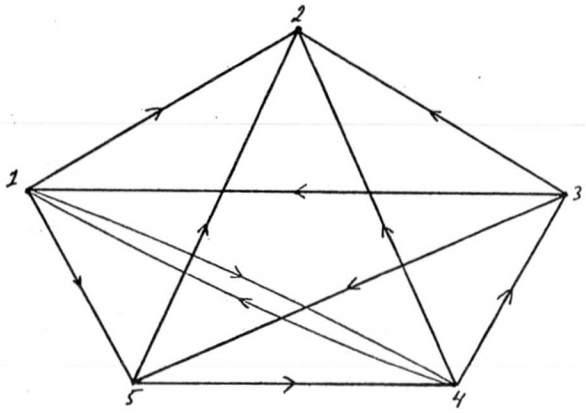

Figure 3. Directed graph where the nodes represent the codes and the arcs the off-diagonal elements ≥ 1 in the matrix of reciprocal ratios of row averages (figure 2).

bound to run into numerical problems. This fact was accentuated by the fact that the dual option of MPSX stopped without finding a feasible solution.

There are several ways to prevent cycling and/or numerical problems. Apart from proper scaling, anti-cycling devices and switching to another mathematical programming package, one which we paid special attention to is one which analyzes the problem formulation. Removing redundant constraints generally reduces the computational effort to solve a problem. In particular, removing weakly redundant constraints (Telgen [1979]) reduces the degree of degeneracy of a solution. Therefore in the latter case both classical and package cycling are less probable. It is unlikely that the phenomenon of cycling exists only by virtue of the presence of redundancy. However, I believe that cycling is not possible if the constraint system is a minimal representation of the feasible region (see Telgen [1979]).

Acknowledgement

The cooperation of T. C. T. Kotiah and D. I. Steinberg in performing one of the experiments is greatly appreciated. Bob Koundenburg (Computer Institute Woudestein, Erasmus University Rotterdam) was very helpful in performing most of the experiments.

References

Dantzig, G. B., Linear programming and extensions, Princeton (1963).

Gass, S. I., Comments on the possibility of cycling with the simplex method, Operations Research 27, No. 4 (1979), pp. 848-852.

Kotiah, T. C. T. and Steinberg, D. I., Occurences of cycling and other phenomena arising in a class of linear programming models, Communications of the A.C.M. 20, No. 2 (1977), pp. 107-112.

Kotiah, T. C. T. and Steinberg, D. I., On the possibility of cycling with the simplex method, Operations Research 26, No. 2 (1978).

Telgen, J., Redundancy and linear programs, dissertation, Erasmus University, Rotterdam (1979).

****

		a	b	K&S
Objective function value		30.693939	30.693970	30.69392
COL01	BS	2.5069958 -1	2.5069952 -1	2.5070020 -1
COL02	BS	8.1723952 -4	8.1723928 -4	8.1724092 -4
COL03	BS	2.8075077 -2	2.8075036 -2	2.8075056 -2
COL04	BS	1.0003197 -1	1.0003197 -1	1.0003298 -1
COL05	BS	0.0000000 +0	0.0000000 +0	3.9764259 -6
COL06	BS	1.3596838 -4	1.3596818 -4	1.3596827 -4
COL07	BS	4.8447028 -4	4.8447051 -4	4.8447529 -4
COL08	BS	5.2973395 -4	5.2971765 -4	5.2971828 -4
COL09	BS	1.0265466 -3	1.0264998 -3	1.0265075 -3
COL10	BS	1.9547403 -1	1.9547427 -1	1.9547282 -1
COL11	LL	0.0000000 +0	0.0000000 +0	---
COL12	LL	0.0000000 +0	0.0000000 +0	---
COL13	LL	0.0000000 +0	0.0000000 +0	---
COL14	LL	0.0000000 +0	0.0000000 +0	---
COL15	LL	0.0000000 +0	0.0000000 +0	---
COL16	BS	1.9905402 -3	1.9905602 -3	1.9905265 -3
COL17	LL	0.0000000 +0	0.0000000 +0	---
COL18	BS	0.0000000 +0	0.0000000 +0	-1.0010399 -9
COL19	BS	3.9164126 -3	3.9164536 -3	3.9163120 -3
COL20	BS	1.1977060 -3	1.1977181 -3	1.1977078 -3

Table 2: Solutions; K&S indicates the solution as reported by Kotiah and Steinberg [1977]

It is interesting to note that in both solutions a and b, x_5 is equal to 0.0. By the argument in section 2 this is impossible, since the solution would be infeasible. However, inspection of the values of the other variables indicate that the value of x_5 is approximately 4×10^{-6}, which may be too small to be printed by MPSX. A similar argument applies to x_{18}.

To conclude, we contacted T. C. T. Kotiah and D. I. Steinberg who graciously agreed to rerun the problem with MPS but without the nonnegativity constraints on the first 10 variables. The problem that originally cycled now was solved in 19 iterations: 14 in phase I, 5 in phase II. The same result obtained when the reduced problem was fed to the unsophisticated code mentioned in the Introduction.

4. Concluding Remarks

The fact that a practical linear programming problem cycled is a rare event, but with the problem of Kotiak and Steinberg [1977] it should not have come totally unexpected. As Gass [1979] noted, the problem is ill-conditioned and

Code \ Problem	1	2	3	4	5	6	7	8
1	F	1.00	1.00	F	0.50	0.67	0.50	F
2	0.67	F	0.60	0.40	F	0.50	0.42	1.00
3	0.25	0.67	F	1.00	1.00	F	1.00	0.71
4	1.00	1.00	0.50	F	0.67	1.00	F	0.50
5	F	0.80	0.75	0.25	F	0.80	0.28	F

Figure 1 Minimum-time ratios of five non-linear optimization codes on eight test problems (F = failure).

Code \ Code	1	2	3	4	5	Row Sums	Normalized	Eigenvector
1	1.00	1.43	0.75	1.00	1.21	5.39	0.21	0.21
2	0.70	1.00	0.84	0.92	0.92	4.38	0.17	0.17
3	1.33	1.19	1.00	0.83	2.01	6.36	0.25	0.24
4	1.00	1.08	1.21	1.00	0.98	5.27	0.20	0.21
5	0.83	1.08	0.50	1.02	1.00	4.43	0.17	0.17
Col. Sums	4.86	5.78	4.30	4.77	6.12	25.83		
Norm. Inv.	0.21	0.17	0.24	0.21	0.17			

Figure 2. Reciprocal ratios of row averages of five non-linear optimization codes on eight test problems (data in figure 1); normalized row sums, normalized inverted column sums, and Perron and Frobenius eigenvector (corresponding to $\lambda_{max} = 5.08$) of matrix of ratios of row averages.

ANNOUNCEMENT OF FUTURE TEST EFFORTS

CALL FOR TEST PROBLEMS

Deterministic Discrete Time Optimal Control
or
Nonlinear Dynamic Problems with Continuous Variables

Mathematical programmers are currently discussing testing methodology for MP-software. One of the major problems in this field is the lack of different types of practical problems that can be used as test examples. In my current work with a code for large scale deterministic discrete time optimal control or nonlinear dynamic problems with continuous variables I feel this lack as an important problem. Therefore, I ask everybody who has practical problems that fit this discription to send me these problems.

Potential suppliers of test problems should send at least the following information:

1. A list of the names and description of all variables in the model divided into a list of optimization variables and a list of exogeneous variables.

2. A list of all constraints in the model. Please write simple lower and upper bounds (both time dependent and time independent) in a separate list.

3. A set of data for the exogenous variables and the lagged optimization variables. If possible, add a set of initial values (feasible or infeasible, time dependent or time independent) for the optimization variables from where the optimization can be started.

4. An objective function, min or max, and an optimization period.

5. If possible, one or more papers that describe the problem, the model, the data, etc.

6. If an optimal solution is known, please include this solution.

In practice we often solve a sequence of related problems. Therefore, a good code should include revise and/or restart capabilities. Problems with more than one objective functions, alternative sets of bounds, different planning periods, different sets of data for the exogeneous variables, and possibly al ternative versions of one or more constraints in the model are welcome. Also, alternative formulations of the same model are welcome. This will make it possible to make empirical studies of model formulation techniques for non-linear models.

The plan is to transform the models into a machine readable form similar to the standard MPS-format for linear programming. The models will later be made available to other researchers in the field.

Please send test problems, inquiries, or comments to: Arne Drud, IMSOR, Building 349, Technical University of Denmark, DK-2800 Lyngby, Denmark.

****

RECENT COMPUTATIONAL TESTING

$$\begin{aligned} &\max\ c^T x \\ &\text{s.t. } Ax \leq b \\ &\quad i^T Ax \geq i^T b \\ &\quad x \geq 0 \end{aligned}$$

where $i^T = (1, \ldots, 1)$. In our experiments we used both the equality (EQ) and the inequality (INEQ) formulation.

The first 11 constraints of the problem have the form

$$\sum_{\substack{j=1 \\ j\neq i}}^{20} a_{ij}x_j - a_{ii}x_i = 0 \quad i = 1, \ldots, 10$$

$$\sum_{j=1}^{10} a_{11,j}x_j = b_{11}$$

with $a_{ij} \geq 0$, $i = 1, \ldots, 10$, $j = 1, \ldots, 20$; $a_{11,j} > 0$, $j = 1, \ldots, 10$ and $b_{11} > 0$. Since $x_j \geq 0$, $j = 1, \ldots, 20$, the first 10 constraints imply that if one of the first 10 x_j's is zero, then all of them are. But this is impossible because of the 11th constraint. Therefore, the nonnegativity constraints on x_j, $j = 1, \ldots, 10$ are redundant.

In fact, using the REDINQ algorithm (Telgen [1979]), it turns out that these are the only redundant constraints in the problem. Removing the redundant constraints (freeing the corresponding values) simplifies the problem considerably. The reduced problem is denoted as either RED-EQ or RED-INEQ. Finally, we applied the REDUCE option of the MPSX problem to both problem formulations. In both cases seven redundant nonnegativity constraints were removed in two matrix passes. Experiments with this problem formulation are indicated by either REDUCE-EQ or REDUCE-INEQ.

3. Experimental Results

All experiments were performed on the IBM 370/168 computer of the Technical University Delft, The Netherlands, using the PRIMAL option of the MPSX package. Table 1 summarizes our results.

	# it phase I	# it phase II	# it total	solution
EQ	12	4	16	a
INEQ	33	4	37	b
RED-EQ	11	7	18	a
RED-INEQ	10	7	17	b
REDUCE-EQ	15	6	21	a
REDUCE-INEQ	11	8	19	b

Table 1: Computational Results

The solutions as found in our experiments and those reported by Kotiah and Steinberg [1977] ccan be compared in Table 2.

RECENT COMPUTATIONAL TESTING

A NOTE ON A LINEAR PROGRAMMING PROBLEM THAT CYCLED[1]

Jan Telgen, Management Science Program[2],
University of Tennessee, Knoxville, Tennessee 37916

Abstract

Kotiah and Steinberg [1977; 1978] reported on an instance of cycling in a practical linear pgoramming problem using IBM's MPS package. We solved their problem a number of times with the MPSX package, using two different formulations and sometimes removing redundant constraints. Results indicate that both the problem formulation and the removal of redundant constraints may improve considerably the performance of the package and prevent the problem from cycling.

1. Introduction

Kotiah and Steinberg [1977;1978] reported on a practical linear programming problem exhibiting the phenomenon known as cycling, when solved with IBM's MPS package. Recently Gass [1979] clarified the situation by distinguishing between classical cycling and computer cycling. Whereas the former is the kind of cycling described in most textbooks, the latter is caused by finite precision computations on real computers. We prefer to use the term package cycling instead of computer cycling, since this kind of cycling is a function of the package used. Gass [1979] already mentioned the fact that the problem of Kotiah and Steinberg [1977] did not cycle when solved with the FMPS/UNIVAC 1108 package. We have similar experience with the MPSX/370 package, but the problem did cycle when solved by an unsophisticated tableau form simplex code.

This note can be seen as an extension of the remarks in Gass [1979]. We report some computational experience with the problem given in Kotiah and Steinberg [1977]. More specifically, we solved that problem using a number of different problem formulations involving both equalities and inequalities and removing none, all, or some of the constraints. The problem formulations are described in section 2, computational results are given in section 3, and some concluding remarks are made in section 4.

2. Problem Formulation

The problem as given by Kotiah and Steinberg [1977] is stated in the form

$$\begin{aligned} \max\ & c^T x \\ \text{s.t. } & Ax = b \\ & x \geq 0 \end{aligned}$$

As Dantzig [1963] noted, an equivalent formulation is

[1]Work performed at Erasmus University, Rotterdam, The Netherlands
[2]Visiting NATO Science Fellow for the Netherlands Organization for the Advancement of Pure Research (ZWO)

ANNOUNCEMENT OF FUTURE TEST EFFORTS

HOW GOOD IS KHACHIAN'S ALGORITHM?

We have been asked to create some order in the chaos caused by Khachian's algorithm and the questions raised by that development. Questions like "does this algorithm really work?"; "what about its numerical stability?"; "does it work with fixed precision computations?"; "which are the improvements that are worthwhile?"; "how good is it in practice?"; "can it solve linear programming problems in a reasonable amount of time?"; how does it compare to the simplex method?"; and, "can it ever beat the simplex method?" tend to get different answers in different papers.

To give some indications on the answers to these and similar questions we will perform a number of computational experiments to determine the most efficient version of the ellipsoidal algorithm. This version will then be used to solve a number of linear programming problems. Results will be compared to those of a commercial LP package.

All tests will be performed according to the COAL guidelines for testing and evaluating mathematical programming software. Results will be presented at the COAL session of the TIMS/ORSA meeting in Colorado (November, 1980) and the COAL symposium in Boulder (January, 1981).

As a result of a recent seminar at IBM, we have been in contact with several people who have suggested ways of improving upon the Gacs-Lovasz version of the algorithm. Anyone wishing to offer further suggestions is welcome to submit ideas to us. We will, of course, give credit to those individuals in our presentation.

If you are interested in collaborating with us in this effort, please contact either of us at the following locations:

John Barrer
School of Management
State University of New York
Buffalo, New York 14214

Jan Telgen
Management Science Program
University of Tennessee
Knoxville, Tennessee 37916

ANNOUNCING A CONFERENCE ON

MATHEMATICAL PROGRAMMING

TESTING AND VALIDATING

ALGORITHMS AND SOFTWARE

at the

National Bureau of Standards

BOULDER, COLORADO

January 5-6, 1981

Mathematical programmers who carry out computational analyses are routinely confronted with a variety of issues for which there is little unanimity. Some examples: What types of test problems should be used--random or hand selected? How can the structure of real-world examples be introduced into problem generators? What evaluation criteria should be used? And how should these criteria be measured? Which algorithms perform best? Developers and users of optimization software will be present to give their view on these topics. Keynote speakers will include Dr. Philip Wolfe, IBM Corporation and Professor Darwin Klingman, University of Texas.

Limited support is available for travel assistance. For further information, contact Professor John M. Mulvey, School of Engineering and Applied Science, Princeton University, Princeton, New Jersey 08544. Conference proceedings will be published.

organized by

Committee on Algorithms (COAL)

Mathematical Programming Society

U.S. National Bureau of Standards

U.S. Department of Energy

RECENT COMPUTATIONAL TESTING

PERFORMANCE RESULTS OF THE SIMPLEX ALGORITHM FOR A SET OF REAL-WORLD LINEAR PROGRAMMING MODELS

Dr. Edward H. McCall, Sperry Univac, P. O. Box 43942,
St. Paul, Minnesota 55164

Extended Abstract

This paper provides performance results using the SPERRY UNIVAC 1100 Series linear programming (LP) product, FMPS, to solve a set of real-world linear programming problems. As such, it provides a data point for the actual performance of a commercial simplex algorithm on real-world linear programming problems, and shows that the simplex algorithm is a linear-time algorithm in actual performance. More specifically, the number of iterations (I) required to optimize an LP model is expressed as the number of constraints (m), variables (n), and nonzeros (nz) raised to a power (a, b, and c, respectively):

$$I = m^a \ , \ I = n^b \ , \ I = (nz)^c \ , \text{ and}$$

the value of that power is determined for the above set of LP problems. The values of the exponents a and b range around one, and the values of the exponent c are less than one. It is also found that the number of iterations required to optimize those models is less than ten times the number of constraints. It is thus concluded that in actual practice the simplex method is a linear time algorithm.

There are also a number of additional results given in this paper. The computer time required to optimize those models is expressed as m, n, and nz raised to a power and the value of that power is determined for the above set of LP problems. Also, for those problems, the ratio of the number of variables to the number of constraints is determined; it is found that the value of that ratio is generally about 2, with a range of 0.79 to 4.7.

Finally, values are determined for the exponents a, b, and c in the following equations:

$$n = m^a \ , \ nz = m^b \ , \ nz = n^c$$

Those values show that for real-world LP models the number of constraints, variables, and non-zeros tend to be related; as models are developed that are larger in any one of these, it will tend to also be larger in the other two.

****

RECENT COMPUTATIONAL TESTING

Group, England), LMDER and LMDIF (MINPACK, Argonne National Laboratories), LSQFDN, LSQFDQ and LSQNDN (NPL Library, National Bureau of Economic Research), NS03A (Harwell Library, AERE, England), SNWT (Jet Propulsion Laboratory), TJMAR1 (Sandia National Laboratories) and ZXSSQ (IMSL Library, Houston).

The paper contains general information about the codes such as the methods implemented, return criteria, and storage requirements and the results of the testing. The emphasis of the testing was on the robustness and reliability of the codes. The set of test problems consisted of 36 problems found in the literature. The design of the testing followed the suggestions of More in [4], e.g., vary the starting value and vary the variable and function scaling. Because of these modifications, each code had 216 problems to solve.

There are many differences among the codes. The most important differences, with respect to the testing, are the differences within the code for determining if a solution has been found. Because of those differences, it was impossible to force the codes to perform completely equivalent tasks in solving the problems. However, the differences and their effect on the performance of the codes are discussed.

The performance of the codes on the test problems is presented in two manners, numerical tables and Chernoff faces. A Chernoff face is a statistical tool used in the analysis of data involving several variables [1]. Tables on efficiency measures, i.e. function evaluation counts and timing, were also included.

The comparison was originally done to advise a committee from several national laboratories on which codes, that solve nonlinear least squares problems, should be adopted for their common mathematical software library. Therefore the conclusions in this paper are concerned with the performance of individual codes with respect to robustness and reliability. A subsequent paper, [3], has been written based on the same data but with the emphasis of the conclusions on the "state-of-the-art" of software in this area, i.e. to determine which areas of the problem are handled well, if one method is "better" than another, and where improvements can be made.

References

1. Chernoff, H., "The Use of Faces to Represent Points in K-Dimensional Space Graphically," Journal American Statistical Association, 68, (1973), pp. 361-368.

2. Hiebert, K. L., "A Comparison of Nonlinear Least Squares Software," Sandia Technical Report, SAND79-0483, Sandia National Laboratories.

3. _____, "An Evaluation of Mathematical Software Which Solves Nonlinear Least Squares Problems," Sandia Technical Report, SAND80-0479, Sandia National Laboratories, submitted to TOMS.

4. More, J. J., Garbow, B. S., and Hillstrom, K. E., "Testing Unconstrained Optimization Software," TM-324 (1978), Applied Mathematics Division, Argonne National Laboratory, Argonne, Illinois.

****

TENTATIVE SCHEDULE FOR COAL CONFERENCE, NBS, BOULDER, COLORADO

Monday (January 5, 1981)

Opening Remarks - Professor John M. Mulvey (Princeton U.) 9:00-9:10

Keynote Speaker - Dr. Philip Wolfe (IBM Corporation) 9:10-10:00
Historical developments in evaluating mathematical programming algorithms

Session Chairs and Topics 10:15-12:00
- Professor Ronald Rardin (Georgia Institute of Technology)
  Design and use of problem generators and hand selected test cases
- Professor Leon Lasdon (U. of Texas at Austin) and Professor Ken Ragsdell (Purdue U.)
  State of nonlinear optimization codes and empirical tests

LUNCH

- Richard Jackson (National Bureau of Standards)
  Approaches to software testing from other disciplines
- Dr. Harlan Crowder (IBM Corporation) 1:30-3:15
  Special topics in testing of MP algorithms (microprocessors, APL, others)
- Dr. Karla Hoffman (National Bureau of Standards)
  Improvement of testing methodologies
- Prof. Darwin Klingman (U. of Texas at Austin) 3:30-5:15
  Advances in network related algorithms

COCKTAIL PARTY (Broker Inn) 5:30-8:00

Tuesday (January 6, 1981)

Keynote Speaker - Prof. Darwin Klingman (U. of Texas at Austin) 9:00-9:45
Computational advances in networks and their transfer to other mathematical programming areas

Session Chairs and Topics 10:00-11:45
- Dr. Richard O'Neill (U. S. Dept. of Energy)
  Evaluation of heuristics
- Prof. Ronald Dembo (Yale University)
  Critique of nonlinear programming experiments and reporting results

LUNCH - COAL Business Meeting (Jackson) (Broker Inn) 11:45-1:15
Future direction and goals of COAL

(to be asigned) 1:15-2:45
miscellaneous topics
- Prof. Roy Marston (U. of Arizona)
  Computational testing of integer programming and combinatorial problems

Prof. John Mulvey (Princeton U.) 3:15-5:00
Panel discussion on establishing a group for testing MP techniques

Note: Each session will include three to four presentations from prominent researchers and practitioners in the relevant areas. An attempt will be made to bring people from a variety of disciplines. A block of rooms has been reserved for conference participants at the Broker Inn (303) 444-3330. Room reservations must be made by November 15, 1980.

Calendar of mathematical programming meetings
as of 1 August 1980

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.

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.

1980

August 4-15: "Generalized concavity in Optimization and Economics", NATO Advanced Study Institute, University of British Columbia, Vancouver, B.B., Canada. Contact: Prof. Siegfried Schaible, Faculty of Business Administration and Commerce, University of Alberta, Edmonton, Alberta, Canada T6G 2G1; Telex 037-2979, Telephone 403-432-5027.

August 28-30: "Bonn Workshop on Combinatorial Optimization", Institute of Operations Research, Bonn, Federal Republic of Germany. Contact: See 1982, August 23-28.

September 15-17: "2nd IFAC Workshop on Control Applications of Nonlinear Programming and Optimization", Oberpfaffenhofen, Federal Republic of Germany. Contact: Dr. Klaus Weil, Institut für Dynamik der Flugsysteme, DFVLR, Oberpfaffenhofen, D-8031 Wessling, F.R.G.; telephone (0 81 53) 2 81.

September 16-18: "6th International Seminar on Algorithms for Production Control and Scheduling", Karlovy Vary, Czechoslovakia. Contact: Ing. Jifi Kral, House of Technology, Gorkeho nam. 23, 11282 Praha 1, Czechoslovakia.

December 10-12: "19th IEEE Conference on Decision and Control", Albuquerque, New Mexico, U.S.A. Submission deadline 31 March 1980. Contact: Prof. Michael K. Sain, Dept. Electrical Engineering, Notre Dame University, South Bend, IN 46556, U.S.A.

1981

January 5-6: "Mathematical Programming: Testing and Validating Algorithms and Software". U. S. National Bureau of Standards, Boulder, Colorado. Organized by the Committee on Algorithms of the MPS, the Bureau of Standards, and the Department of Energy. Contact: Dr. Richard H. F. Jackson, Center for Applied Mathematics, National Bureau of Standards, Washington, D.C. 20234; telephone 301-921-3855.

January 26-31: "Mathematische Optimierung", Mathematisches Forschungsinstitut Oberwolfach, Oberwolfach, Federal Republic of Germany. Contact: Institut für Ökonometrie und Operations Research (see 1982, August 23-28).

April 6-8: "International Congress on Mathematical Programming", Rio de Janeiro, Brazil. Contacts: Professor R.W. Cottle, Dept. Operations Research, Stanford University, Stanford, CA94305, U.S.A.; Professor Milton Kelmanson, Pontificia Univesidade Catolica, Departamento de Engenharia Eletrica, Rua Marques de Sao Vicente, 225, Rio de Janeiro, R.J., Brazil; Professor B. Korte, Bonn (see 1982, August 23-28).

May (preliminary notice): "Optimization Days" (normally a two-day meeting, held at a university in Montr<al, Canada. Professor Jean-Marc Rousseau Departement d'informatique et de R. O. Universite de Montreal C.P. 6128, Succursale "A" Montreal, P.Q., H3C 3J7

References

[1] Denardo and Fox, "Shortest-Route Methods: 1. Reaching, Pruning, and Buckets," Operations Research, 27, 1 (1979) 161-186.

[2] R. Rial, F. Glover, D. Karney, and D. Klingman, "A Computational Analysis of Alternative Algorithms and Labeling Techniques for Finding Shortest Path Trees," Networks, 9 (1979) 215-248.

[3] J. Elam, D. Klingman, and J. Mulvey, "An Evaluation of Mathematical Programming and Minicomputers," Research Report CCS 313, Center for Cybernetic Studies, The University of Texas at Austin. To appear in European Journal of Operations Research.

[4] M. Florian, S. Nguyen, and S. Pallottino, "Dual Simplex Approach to Finding All Shortest Paths," Publication #39, Transportation Research Center, University of Montreal, October 1979. To appear in Networks.

[5] J. Gilsinn and C. Witzgall, "A Performance Comparison of Labeling Algorithms for Calculating Shortest Path Trees," NBS Technical Note 772, U. S. Department of Commerce, 1973.

[6] F. Glover and D. Klingman, "A Survey of Network Models, Methods, and Applications," presented at the XXIV International Meeting of the Institute of Management Science (TIMS), Honolulu, Hawaii, July 10-13, 1979.

[7] D. Klingman, J. Mote, and D. Whitman, "Improving Flow Management and Control Via Improving Shortest Path Analysis," Research Report CCS 322, Center for Cybernetic Studies, The University of Texas at Austin, 1978.

[8] D. Klingman and J. Mulvey, "Applicability of Shortest-Path and Minimum-Cost Flow Network Algorithms for Minicomputers and Micro-Processors," Proceedings of the 9th International Mathematical Programming Symposium, Budapest, August 23-27, 1976.

****

TESTING NONLINEAR LEAST SQUARES SOFTWARE†

Kathie L. Hiebert, Numerical Mathematics Division 5642,
Sandia National Laboratories*, Albuquerque, New Mexico 87185

This paper [2] reports the results of comparing 12 readily available FORTRAN codes which solve unconstrained nonlinear least squares problems. The codes in the comparison are E04GAF and E04FAF (NAG Library, Numerical Algorithms

†This article sponsored by the U. S. Department of Energy under Contract DE-AC04-76DP00789.
*A U. S. Department of Energy facility.

3. Testing

The testing focused on the computer implementation of shortest path algorithms in a high level computer language. No implementation enhancements were tested which could be derived from assembly language coding. Additionally, our testing was restricted to solving 2000 problems involving only two distinct topology types with three different arc length ranges (1-100, 1-200, 1-2000). Consequently, caution should be exercised in undertaking to extrapolate the empirical results beyond the classes of problems considered and beyond the particular codes tested.

4. Findings

Three codes dominated all others on the test problems, one label-correcting code and two label-setting codes. The best label-correcting code was decidedly superior to all alternatives on grid networks and sparse random networks. The two best label-setting codes turned out superior for denser problems, though the range of arc length coefficients affected the outcome also. For instance, a label-setting code surpassed the label-correcting code when the average number of arcs per node exceeded 10, provided arc lengths were in the 1-200 range. When arc lengths were in the 1-10,000 range, the label-setting code became superior only when the average number of arcs per node exceeded 15.

5. Availability of the Best Computer Codes

Computer listings for the best procedures, all in FORTRAN, can be obtained by writing either of the authors.

6. Additional Recent Results

Subsequent to the study of [2], additional shortest path studies have been undertaken which have removed some of its limitations. In particular, the restriction to all in-core algorithms and large computer main frames has been eliminated in studies by Klingman, Mote, and Whitman [7], Elam, Klingman, and Mulvey [3], and Klingman and Mulvey [8], which extended these algorithms to "in-core out-of-core" implementations and mini-computers. Further, studies have now appeared on additional shortest path methodology. The first, by Denardo and Fox [1], develops new "multiple bucket" algorithms that are projected to be efficient for extremely large problems, though not for problems of sizes tested in their study. (Extrapolation of solution times for the multiple bucket algorithms suggested the possibility that the multiple bucket methods would dominate for some larger problem size.) Another study by Florian, et al [4] focuses on advanced dual methods for problems of finding shortest paths between all pairs of nodes (in contrast to finding shortest paths from a single source to all others), and reports promising gains in efficiency for these problems. Finally, more recent investigations by Glover and Klingman [6] utilize a thresholding procedure for selecting nodes to scan that has produced some surprising and dramatic results. In particular, limited testing indicates that a label-correcting code which embodies this technique is superior to all previous label-correcting and label-setting codes on all problem topologies.

CANADA Contact: Prof. Alain Haurie, Service de l'enseignement des Méthodes Quantitatives, École des Hautes Études Commerciale, 5255 avenue Decelles, Montréal, Québec, Canada H3T 1V6; télephone 514-343-3801.

July 13-24: "NATO Advanced Research Institute on Nonlinear Optimization", Cambridge, England. Contact: Professor M.J.D. Powell, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Silver Street, Cambridge CB3 9EW, England. Sponsored by the MPS.

1982

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.)

ANNOUNCEMENT

THE INSTITUTE OF MANAGEMENT SCIENCES
OPERATIONS RESEARCH SOCIETY OF AMERICA

Special Interest Meeting

APPLIED PROBABILITY-
COMPUTER SCIENCE,
THE INTERFACE

January 5 - 7, 1981

FLORIDA ATLANTIC UNIVERSITY
BOCA RATON, FLORIDA

Topics to be covered include:

- Probabilistic Models in Performance Analysis of Computer Systems and Communication Networks
- Networks of Queues
- Probabilistic Analysis of Databases
- Probabilistic Scheduling
- Probabilistic Aspects of Simulation
- Probabilistic Analysis of Algorithms
- Computational Aspects of Applied Probability

For information contact:

Teunis Ott
Bell Telephone Labs
HP Rm. 1A332
Holmdel, New Jersey 07733
(201) 561-7100, x2107

RECENT COMPUTATIONAL TESTING

The outstanding study by Gilsinn and Witzgall [5] pioneered in demonstrating the importance of computer implementation technology for shortest path algorithms by showing that its use could reduce solution times for shortest path problems from one minute to slightly more than one second, using the same general shortest path algorithm, computer and compiler. The Gilsinn and Witzgall study also made important contributions toward a unified structure for describing such algorithms. Our study [2] has undertaken to build on the foundations provided by Gilsinn and Witzgall, evaluating an expanded range of solution procedures, and further demonstrating the importance of computer implementation technology by the exposition of new procedures superior to those previously documented. These new procedures have made important use of ideas proposed by Dr. Ellis Johnson for streamlining simplex-type shortest path calculations and of ideas proposed by Professor U. Page for growing and processing lists of scan-eligible nodes from both ends.

2. Experimental Design

Alternative implementation methods have been evaluated in this study by solving a diverse set of randomly generated shortest path problems using the same computer (a CDC 6600), the same compiler (a FORTRAN RUN compiler), and executing the codes during time periods when the demand for computer use was comparable. Further, all of the codes were implemented by the same systems analyst and no attempt was made to exploit any of the unique hardware characteristics of the CDC 6600.

Even with these safeguards, minor differences between the solution times of any two codes for a single test run of each must be regarded as being of questionable significance. For this reason, each test problem was solved 100 times (i.e., for 100 different roots) and the average solution time reported. Each code made use of a real-time clock routine supplied by CDC. This routine can be employed using a FORTRAN subroutine call and is generally accurate to two decimal places. The reported times include only the elapsed time after input of the shortest path problem and prior to output of its solution. This includes the time required to initialize the function arrays.

The problem set consists of shortest path problems from two distinct topological groups, rectangular grid networks and random networks. The grid network test problems, whose structure occurs in practical applications of transportation planning, were examined for different degrees of rectangularity for problem sizes involving 2,500 nodes and in the neighborhood of 10,000 arcs. Arc lengths were allowed to vary between 1 and 100 for half the problems tested, and between 1 and 10,000 for the other half. The random network test problems all had 1000 nodes and contained either 5000, 10000, 15000, 20000, 25000, or 30000 arcs. For each of these problem sizes, two problems were generated, one with arc lengths between 1 and 200 and the other with arc lengths between 1 and 10,000. In all cases the arc lengths were randomly selected using a uniform probability distribution. (To provide researchers with reproducible benchmarks, interested readers can obtain from the authors FORTRAN listings of the problem generators and the two computer codes found to be the best in this study.) A limitation of this design is that it does not include real-life large networks.

RECENT COMPUTATIONAL TESTING

PRECIS OF A COMPUTATIONAL ANALYSIS OF SHORTEST PATH ALGORITHMS

Fred Glover, Professor of Management Science, U. of Colorado,
Boulder, Colorado 80309

Darwin Klingman, Professor of Operations Research & Computer Sciences
BEB 608, University of Texas, Austin, Texas 78712

1. Introduction

The importance of shortest path analysis in numerous quantative transportation and communication models has caused a proliferation of shortest path algorithms in the literature. However, there are only a handful of general methods for solving shortest path problems by which we mean problems of finding the shortest paths from a single node to all other nodes in a directed network. Each general algorithm has a number of subalgorithms within it. These are designed to handle special subproblems or sets of operations such as finding the minimum of a set, breaking a loop, reconnecting subtrees, carrying out computations over the nodes and arcs of subtrees, etc. The many different ways developed to handle these subproblems, unfortunately, have often been referenced as different algorithms rather than as variants of the small class of general algorithms. The study of [2], whose highlights are recapitulated (and supplemented) here, shows how to organize many of these methods into algorithm/subalgorithm classifications, with special emphasis on describing their operations from a computer implementation standpoint. The study also demonstrates the perhaps surprising result that these algorithms can be viewed as special cases of the primal simplex method. Finally, the study reports the outcomes of extensive computational testing that we have recently extended to identify a more effective subclass of shortest path algorithms.

The historical development of the many shortest path "algorithms" has paralleled and contributed to the growth of an important interface between mathematics and computer science, called computer implementation technology. This technology, which only recently has been recognized as a major discipline in its own right, involves the design of special procedures to carry out subalgorithms of a general method efficiently on a digital computer. The slowness in some quarters to appreciate the importance of this area has no doubt contributed to the past confusion in the shortest path literature between generalized and special methods. Computer implementation technology requires research to determine: (1) the kinds of information to keep on hand for executing certain operations most effectively, (2) the kinds of data structures in which to express this information, and (3) the actual methods for processing these data structures to make the desired information available when it is needed. Effective use of such research further involves design by feedback, iteratively amending and integrating component procedures by reference to computational analysis and performance.

CALL FOR PAPERS

Mathematical Programming Study
on
Matrix Generation, Report Writing, and
Computer-Assisted Analysis

Edited by:

Dr. Harvey J. Greenberg
Energy Information Administration
Department of Energy
1200 Pennsylvania Avenue, N. W.
MS 4530
Washington, D. C. 20461
(Phone: (202) 633-9790)

This special study on MG/RW/CAA considers computer aides for model management in mathematical programming. Large-scale linear programming models, and extensions to nonlinear and integer forms, use modern methods of computer science to develop, use, and maintain systems. Papers are hereby solicited, which conform to the standards of Mathematical Programming, that pertain to these support functions, apart from optimization. Topics include:

- Information structures
- Language design
- Algorithms
- Performance evaluation
- In-depth surveys
- Diagnostic analysis
- Interactive processing.

Please send three copies of your manuscript to Harvey Greenberg (see address above) and one copy to the Editor-in-Chief of Mathematical Programming:

Dr. Richard W. Cottle
Operations Research Department
Stanford University
Stanford, California 94305

THE MATHEMATICAL PROGRAMMING SOCIETY

ENROLLMENT

I hereby enroll as a member of the Society for the calendar year 1980.

PLEASE PRINT:

Name_____

Mailing address_____

_____

_____

_____

My subscription to *Mathematical Programming* is for my personal use and not for the benefit of any library or other institution.

Signed_____

The dues for 1980 are:

42 Dollars (U.S.A.)
20 Pounds (U.K.)
68 Francs (Switzerland)
178 Francs (France)
76 Marks (Fed. Rep. Germany)
84 Guilders (Netherlands)

Please send this application with your dues to

The Mathematical Programming Society
c/o The International Statistical Institute
428 Prinses Beatrixlaan
2270 AZ Voorburg, Netherlands

EDITOR'S COLUMN

The Committee on Algorithms has been rather active the past six months. Proceedings of the special session jointly sponsored by COAL and the Computer Science Technical Section of ORSA has been published in the April 1980 issue of SIGMAP. Eight papers on "Recent and Future Developments of Math Programming Systems" appear in that issue. Copies of this issue are available to "Friends of COAL," free of charge, while supplies last. Any requests for this issue should be sent to Richard P. O'Neill, Department of Energy, Div. of Oil and Gas Analysis, Washington, D. C. 20461.

In the near future, all "friends" of COAL will be receiving a survey requesting information about MP codes. This survey will be used to compile a list (complete with characteristics) of the mathematical programming software available throughout the world. The Committee intends to sort and collate all information received and to update regularly this information. We believe this information is useful to a wide cross-section of the scientific community and intend to maintain this information service in the future. The results of such an effort can only be successful if we have cooperation from the entire mathematical programming community. So please help us by completing and returning this survey form.

On January 5 - 6, 1981 a conference on Testing and Validating MP Algorithms and Software will be held in Boulder, Colorado. A tentative schedule of that conference can be found elsewhere in this newsletter.

Finally, Doug Shier has been invited to chair a session on computational testing of mathematical programming software at the "Optimization Days, 1981" meeting to be held in Montreal on May 13-14, 1981. If anyone is interested in presenting recent computational results, they should contact Doug Shier, National Bureau of Standards, Center for Applied Mathematics, Washington, DC, 20234. Short abstracts of papers are due by September 14, 1980.

In closing, I encourage each of you to inform me of research being done by you or your colleagues in the area of development and testing of optimization software. The deadline for the next issue is November 15, 1980. Please let me hear from you.

COMMITTEE ON ALGORITHMS of the
MATHEMATICAL PROGRAMMING SOCIETY

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.

THIRD CLASS

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

T. Steihaug
School of Organiza. & Mgmt
56 Hillhouse Ave.
New Haven, Ct 06520

Mathematical Programming Society
Committee on Algorithms Newsletter

August 1980 — Karla L. Hoffman, Editor

Contents: