A GENERATOR FOR RANDOM SAMPLES FROM A CLASS OF MULTI-PRODUCT INGREDIENT-MIX LINEAR PROGRAMMING PROBLEMS

by
James K. Ho
and
Rangarajan Sundarraj

Management Science Program
College of Business Administration
University of Tennessee
Knoxville, TN 37996-0562

October, 1986

ABSTRACT

In the continuing development of new or improved algorithms for Linear Programming, the need for empirical testing with real problems is well recognized. For the purpose of experimental design, it has become increasingly important to have random samples from given classes of models that are statistically significant and yet physically meaningful. This paper presents a generator for a class of multi-product ingredient-mix LP's which has the block-angular structure. The dimensions of the problems and the coefficients of the constraints can be randomized automatically over ranges based on real data. The underlying model as well as features of the generator are described. The software is available on both mainframes and microcomputers and will be made available to the research community.

Keywords: Linear Programs, Matrix Generator, Computational Testing.

1. Introduction

It is well known that the Simplex Method for linear programming works "surprisingly" well on problems arising from most real applications. This is based on the observation that while it is possible for the algorithm to take an exponential number of iterations in terms of **m**, the number of constraints, in practice this number is often just **km**, where **k** is typically between 1 and 5. For this reason, in the comparative testing of new algorithms for LP, behavior in the worst case is generally considered to be of much less importance than behavior in practical cases. While the need for test problems arising from actual or at least realistic applications is recognized and efforts have been made to compile them (see e.g. [1]), real problems usually represent isolated cases. Even for a given type of model, one would need a more significant sample size in order to investigate "average" behavior. It should therefore be useful to have generators for physically meaningful yet statistically significant random samples of real applications.

In the following, we describe a class of multi-product ingredient-mix linear programs, as examplified by an optimization model for the meat processing industry. MEATGEN, a generator of random samples of LP problems from this model is then introduced. A simple example of how such random samples are produced and used in a numerical experiment is also given.

2. The Single Product Ingredient Mix LP

The LP formulation for processed meat is a model to determine the least cost combination of ingredients to meet specified product composition. Our example is for the processing of sausages according to requirements listed in Table 1. The costs and compositions of the ingredients are given in Table 2. Note that for simplicity, only 5 ingredients are included in this example. Data for 15 ingredients ([3], [4]) are built into the generator to be described later.

Finished weight	=	100 lbs.
Maximum fat	=	30 %
Minimum protein	=	10 %
Minimum moisture	=	50 %

Table 1. Product Requirement.

Serial #	Ingredient	Cost	% Fat	%Protein	%Moisture
1	FC Bull	0.95	11.8	19.1	67.9
2	FC Cow	0.94	15.1	18.0	66.1
3	PK Jowls	0.37	70.0	6.3	23.4
4	BF Trmgs-85	0.84	20.0	16.8	62.0
5	Water	0.001	0.0	0.0	100.0

Table 2. Composition of Ingredients

Let x_i be the weight in lbs of ingredient i used in the sausage. The LP is then

$$\begin{aligned} \text{Minimize} \quad & 0.95 x_1 + 0.94 x_2 + 0.37 x_3 + 0.84 x_4 + 0.001 x_5 \\ \text{subject to} \quad & x_1 + x_2 + x_3 + x_4 + x_5 = 100 \\ & 11.8 x_1 + 15.1 x_2 + 70 x_3 + 20 x_4 \leq 3000 \\ & 19.1 x_1 + 18.0 x_2 + 6.3 x_3 + 16.8 x_4 \geq 1000 \\ & 67.9 x_1 + 66.1 x_2 + 23.4 x_3 + 62 x_4 + 100 x_5 \geq 5000 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0. \end{aligned}$$

(9) *McCORMICK, G.P. (1978)*

'Large-scale non-linear programming', Technical Paper Serial T-378, Institute of Management Science and Engineering, The George Washiangton University.

(10) *McCORMICK, G.P. (1983)*

'Non-linear programming: Theory, algorithms and applications', John-Wiley.

(11) *MIELE, A. & GONZALEZ, S. (1978)*

'On the comparative evaluation of algortihms for mathematical programming problems', in non-linear programming 3, pp. 337-359. Edited by Olvi L. Mangasarian, Robert R.Meyer and Stephen M. Robinson. Academic Press, New York.

(12) *MORE, Jorge J., GARBOW, Burton S. & HILLSTROM, Kenneth E. (1978)*

'Testing Mathematics Division, Argonne National Laboratory, Illinois 60439.

(13) *MORE, Jorge J. & COSNARD, Michel Y.*

'Numerical Comparison of 3 Non-linear Equation Solvers.'

(14) *RAGSDELL, K.M. (1974)*

'On some experiments which delimit the utility of Non-linear Programming Algorithms' School of Mechanical Engineering, Purdue University, West Lafayette, Indiana.

(15) *SHAYAN, M.E. (1979)*

'Doctoral Dissertation', Department of Operations Research, The George Washington University, W-DC.

(16) *STEWART, G.W. (1973)*

' Introduction to Matrix Computations', Academic Press, New York.

(17) *U.S. DEPARTMENT OF HEALTH, EDUCATION AND WELFARE (1964)*

'Smoking and Health', Report to the Surgeon General of the Public Health Service.

TABLE 2

NUMBER OF FUNCTIONAL EVALUATIONS REQUIRED TO SOLVE PROBLEM 4.1

Main Algorithm	Algorithm Solving Step-Size Problem	FUNCTIONAL FORM				
		$f(x)$	$\nabla f(x)$	$\nabla^2 f(x)$	$S^T \nabla_x f(x)$	$S^T \nabla^2_{xx} f(x) S$
Rank One	Secant	10	10	--	53	--
Rank One	Newton	10	10	--	43	43
D.F.P.	Secant	10	10	--	47	--
D.F.P.	Newton	10	10	--	44	44
B.F.G.S.	Secant					
B.F.G.S.	Newton	10	10	--	43	43
Newton	Armijo	24	10	10	--	--
Newton	Secant	14	7	7	34	--
Newton	Newton	14	7	7	35	35
Broyden	--	60	60	--	--	--
Generalised Secant*	Secant	17	17	--	43	--

* See notes under Table 1.

3. The Multi-Product Ingredient-Mix LP

A typical production planning problem in the meat processing industry involves many products for which the optimal ingredient mix can be modeled as an LP similar to the one formulated above. For given supplies of the ingredients, the multi-product optimization problem becomes an LP with the block-angular structure (Figure 1). Each coupling constraint states that the total requirement of an ingredient for all products must not exceed the supply.

To formulate the general model, let

I	=	the number of ingredients;
P	=	the number of products;
N	=	the number of nutrients;
p_{ij}	=	% composition of nutrient j in ingredient i;
r_{kj}	=	% requirement of nutrient j in product k;
c_i	=	unit cost of ingredient i;
a_i	=	supply of ingredient i;
d_k	=	demand for product k;
x_{ik}	=	amount of ingredient i (in same unit as d_k) in product k.

Then the multi-product ingredient-mix LP is to

$$\text{minimize} \quad \sum_{k=1}^{P} \sum_{i=1}^{I} c_i x_{ik}$$

$$\text{subject to} \quad \sum_{k=1}^{P} x_{ik} \le a_i, \ i=1,\dots,I: \qquad \text{[Supply];}$$

$$\sum_{i=1}^{I} p_{ij} x_{ik} \le r_{kj} d_k, \ k=1,\dots,P;\ j=1,\dots,N: \qquad \text{[Composition];}$$

$$\sum_{i=1}^{I} x_{ik} = d_k, \ k=1,\dots,P: \qquad \text{[Demand];}$$

$$x_{ik} \ge 0,\ i=1,\dots,I;\ k=1,\dots,P.$$

The model has the following dimensions.

Number of coupling rows	=	I
Number of other rows	=	PN
Number of structural columns	=	P I

4. Random Sampling from the General Model

Randomization over physically meaningful ranges will be considered for the following model parameters:

i) the nutrient requirements of the products;

ii) the composition of the ingredients;

iii) the demand for the products;

iv) the supply of the ingredients.

The purpose is to simulate a variety of realistic production situations which can be used as numerical test problems of desired dimensions and sample size. In order to facilitate problem generation, all data ranges have been built into the model and only a few controls need to be set by the user of a generator program.

REFERENCES

(1) *AL-KHAYYAL, FAIZ, A. (1976)*

'Estimates of the equivalent number of binary operations required for the numerical evaluation of some elementary functions of single variables', Technical Memorandum Serial TM-66184, Institute of Management Science and Engineering, The George Washington University.

(2) *CORNWELL, Larry W., HUTCHINSON, Patricia A., MINKOFF, Michael, & SHCULTZ, Hilbert K. (1978)*

'Test problems for constrianed non-linear mathematical programming algorithms', TM-320, Applied Mathematics Division, Argonne National Laaboratory, Illinois 60439.

(3) *CROWDER, H.P., DEMBO, R.S., & MULVEY, John M. (1978)*

'Reporting Computational experiments in Mathematical Programming', Mathematical Programming 15.

(4) *GHAEMI, A. & McCORMICK, G.P. (1979)*

'Symbolic factorable SUMT: What is it? How is it used?' Technical Paper Serial T-402, Institute of Management Science and Engineering, The George Washington University.

(5) *GILSHINN, J., HOFFMAN, K., JACKSON, R.H.F., LEYENDECKER, E., SAUNDERS, P. & SHIER, D. (1977)*

'Methodology and Analysis for Comparing Discrete Linear L1 Approximation Codes', Commun. Statist. - Simula. Computa, B6(4) 339-413.

(6) *HIMMELBLAU, David M. (1972)*

'Applied Non-Linear Programming', McGraw-Hill Book Co. Highstown, New Jersey.

(7) *JACKSON, Richard J.F. & MULVEY, John M. (1978)*

'A critical review of Comparison of Mathematical Programming Algorithms and Software (1953-1977).

(8) *McCORMICK, G.P. (1974)*

'A mini-manual for the use of the SUMT computer program factorable programming language', Technical Report SOL 74-15, System Optimization Laboratory, Department of Operations Research, Stanford University.

TABLE 1

SUMMARY OF AMOUNT OF WORK REQUIRED TO SOLVE THE UNCONSTRAINED PROBLEM 4.1.

Class	No.	Main Algorithm	Algorithm Solving Step Size Problem	No. Basic Functions	Total Number of Basic Operations				Number of Iterations
				e^x	+	-	x	÷	
1. Quasi-Newton	1	Rank One	Secant	2368	15060	10747	50955	14335	10
	2	Rank One	Newton	1728	14251	7809	46759	10482	10
	3	D.F.P.	Secant	1856	11804	8410	39987	11328	10
	4	D.F.P.	Newton	1760	14383	7954	47724	10756	10
	5	B.F.G.S.	Secant		*does not converge*				
	6	B.F.G.S.	Newton	1728	14278	7809	46975	10644	10
2. Newton	7	Newton	Armijo	768	5648	3232	16902	4688	9
	8	Newton	Secant	1568	10464	6988	34456	9468	7
	9	Newton	Newton	1600	13178	7116	42939	9686	7
3. Broyden	10	Broyden	--	1952	14510	10577	42262	12423	60
4. Secant	11	Generalized Secant*	Secant	2080	13351	9788	44809	12587	13
n+1 points generated by Broyden	–	–	--	128	725	350	1470	40	–

* Generalised secant is a new version of Wolf 's secant using step-size as well [14].

5. MEATGEN: the Matrix Generator for the Model

This is written in Pascal and available in versions for both mainframes and microcomputers. It is available to the research community at cost of material and handling. The logic of the generator is summarized in pseudo code below. The actual code provides the options of creating the LP data file in either industry standard MPS format or in the DECOMP format used in decomposition codes described in [2]. Since the latter may not be of general interest, it will be omitted from further discussion. Users who are interested in experimentation with structured LP's may obtain detailed information on the DECOMP option from the authors.

```
Program MEATGEN;
Procedure INPUT;
    Read ingredient compositions;
    Read product specifications;
    Read ingredient costs;

Precedure PARAMETERS:
    Read number of ingredients;
    Read ingredient supplies;
    Read product demands;

Precedure MPSFILE;
    Define output file;
    Write NAME heading;
    Write ROWS heading;
    Write row types and row names;
    Write COLUMNS heading;
    For each product, write nonzero coefficients;
    Write RHS heading;
    Write ingredient supplies;
    Write product demands;
    Write ENDATA heading;

Main routine:
    INPUT;
    PARAMETER;
    MPSFILE.
```

The generator is run interactively. The user needs only to select the number of ingredients between 6 and 15, specify a target problem size (in terms of total rows), and the sample size. The program will determine the other parameters so as to generate a set of LP's with row dimensions closest to the target size. Since we are particularly addressing microcomputer applications, the maximum row size is currently set to 300, which corresponds to 71 products. Larger problems can of course be created. However, since the underlying model is so simplistic, extension by increasing the number of products alone will make it more and more unrealistic as a real problem.

6. An Example Experiment with a Random Sample

To illustrate the potential applications of generating random samples from a class of real LP problems, we conclude with a simple experiment. Ten problems were generated for each of five problem sizes: 43, 58, 75, 95 and 115 rows respectively. All the problems were solved and the number of simplex iterations required was recorded. The means and standard deviations of the results are plotted in Figure 2. Even for such a trivial exercise, the need for some statistical control over test problems seems to be borne out by the observation of significant variations among random cases, especially with increasing problem size.

For the computer used, we can find the work done in each algorithm by multiplying the total number of basic operations (+, -, x, +) by (2.4, 2.4, 13.35, 7.25) respectively. Table 1 also illustrates how confusing the result of looking at number of iterations and number of function evaluations could be. For example, although in Broyden's method (No.10), 60 iterations were required against 10 iterations in No.1. The actual amount of work performed in No.10 is less than what is done in No.1, and No.7 with 9 iterations. If we look at the number of functions/derivatives evaluated, the situation gets worse because it changes to a multidimension case, as shown in Table 2. There is no simple (if any) way of comparing algorithms on this basis even when problem, representation is standardized. The way counting method concludes is to convert the number of basic operations to time by and rank the algorithms accordingly.

Obviously, ranking of algorithms will be different for each problem solved. It seems that a tremendous number of test problems are required to be solved and proper statistical analysis techniques (or any other way) must be investigated to come up with a general ranking of the algorithms. This was not, however, the purpose of the present paper.

The third type of work involves evaluation of all functional forms used in the algorithm including the main functions and their derivatives of any kind and order.

Although functions seem to be non-ending in shapes, perhaps amazingly they are composed of a limited number of basic functions of one variable, except for a few called "non-factorable". Each evaluation of the basic functions, such as e^x, $\ln x$, x^n, $\sin x$, could again be counted in the entire program. On the other hand, evaluation of such basic functions are done by approximating formulas, Taylor's series for example, which consists of only basic operations. This enables us to reduce the measure of the amount of work to evaluate any function/derivative to an equivalent number of basic operations or time. [1] investigates the equivalent number of basic operations required to evaluate each basic function once and also converts these to the time required for a given computer. Although it will depend on the computer system used, for a given ccomputer, it will be completely known and the same for all algortihms run. Then the problem of computer load will be of no importance in comparison of algorithms, since the time to perform a basic operations is fixed for any computer.

3.1 *Standardization.*

At this point the methodology is complete by asking to count all basic functions, convert them to basic operations, get the sum of all basic operations and convert it to time for a given computer. However as it stands it may fail to produce a convincing result as it neither employs a standard functional representation nor it distinguishes between amount of storage used.

These characteristics are achieved by application of the Factorable Programming Language developed by McCormick [4], [8], [10]. This language is applied to a large class of functions called "Factorable". A factorable function is formed sequentially by terms which are functions of single variable or product of such functions. The variable of each function could be a previously defined term. All the user is required to do is to input the original functions with a special format. Then the functions and their derivatives of first/second order are evaluated with no need for any extra information. This makes the whole work programmer independent.

Additional properties of such representation are: firstly the storage requirments for derivatives is reduced to its minimum and secondly it reduces and standardizes the work required to perform algorithmic calculations involving gradients, matrices (Hessian) by applying dyadic rather than explicit forms.

[15] extends these capabilities to evaluate the directional derivatives in a standard way. These features are nonexistant in the other comparison methods.

The resulting conclusion is that application of Factorable Programming makes the counting methodology feasible and effective for comparing algorithms. Next its application is demonstrated through an example.

4. SOME RESULTS AND CONCLUSIONS.

To show how the method works, the following problem, formulated by G.P. McCormick, for the Smoking and Health Problem Report [17],

$$\min \quad \sum_{i=1}^{16} \left[y_i - 1 - (a_1 x_{i2} + a_2) \left\{ 1 - e^{-a_3 x_{i1}} \right\} \right]^2$$

$$a_j > 0$$
$$j=1,2,3$$

was solved by several algorithms. Result shown in Table 1 is taken from [15] (table x).

References

[1] J.K. Ho and E. Loute, "A set of staircase linear programming test problems", *Mathematical Programming* 20 (1981) 245-250.

[2] J.K. Ho and E. Loute, "An advanced implementation of the Dantzig-Wolfe decomposition algorithm for linear programming",*Mathematical Programing* 20 (1981) 303-326.

[3] Meat Processing, September 1973.

[4] United States Department of Agriculture, Human Nutrition Information Service, Agricultural Handbook Vol. 8-10 (Revised) August 1983.

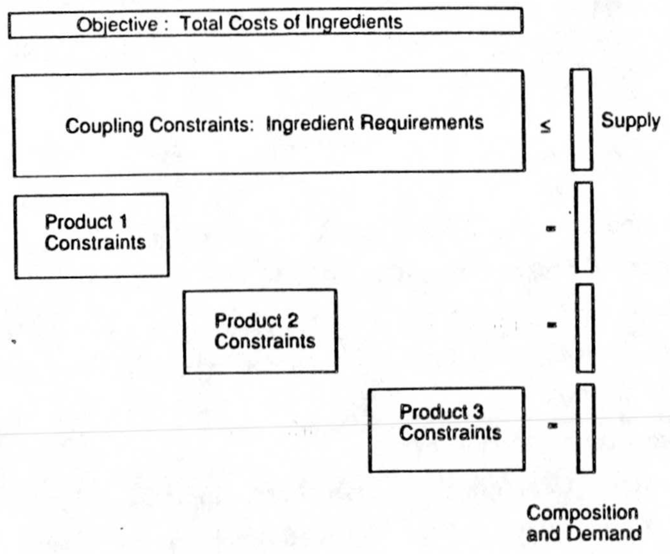

Figure 1. Block-angular Structure of the Model

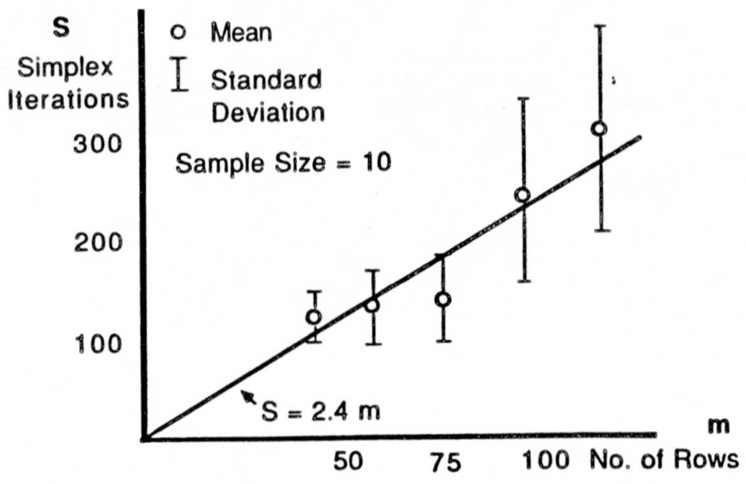

Figure 2. An example experiment.

The amount of work is not the same for each iteration in different algorithms. This is obvious, since one algortihm uses only functional evaluations while the other may use second derivative and all the complications of matrix manipulation. This can be seen even among the iterations of one algorithm, where in one iteration a few steps of line search is sufficient and in the next, many times more may be required. For large problems even the programming style might affect the total work considerably. In this sense, the rate of convergence is not a good measure for comparing algortihms either.

(d) *Storage Requirements.*

This factor is rarely considered in comparing algorithms because it can not be put into numbers for each algorithm, in a consistent manner. However, it is sometimes very critical when the problem is large, and some computers evaluate cost, based on storage use only.

In conclusion, factors currently used to compare algorithms are generally not accurate, though they might provide enormous information, possibly enough for some conclusions.

In addition to all the above discussions, some other important factors and features of a comparison method are either left-out or under-emphasized in most of the works. One factor is the standardization of functional representation, resulting independence to programmer to a much higher degree, and another factor is the actual number representation of the computer itself (eg. to distinguish between single and double precision), which affects both storage requirements and time considerably. For example, evaluation of *ln* x takes 85 micro seconds (ms) in single precision, against 145 ms in double precision on the same computer. Next, a new approach will be introduced which will overcome the previously mentioned difficulties.

3. *A NEW METHODOLOGY FOR COMPARING MATHEMATICAL PROGRAMMING ALGORITMS .*

Inadequacy of present comparing methods is mostly due to the fact that they all utilize an indication of the amount of work done by an algorithm to solve a problem, rather than the actual amount of work performed, the measure taken in this work for comparison.

To measure the total amount of work we must recognise and separate the different work involved in solving a problem, and try to measure them quantitatively for the whole solution process, rather than just one iteration as in general amount of work is not the same for all iterations.

Each mathematical programming algorithm consists of three types of work:

1. Administrative works such as I/O matirx and form generation, etc.
2. Numerical calculations other than functional evaluation such as operations on matrices, etc.
3. Evaluation of functions.

The first type is arguably not related to the speed as it could be imposed on any algorithm according to the user's taste or management's expectations/requirements. So it is not considered here.

The second type simply refers to the basic operations viz. addition, subtration, multiplication and division of numbers. To quantify this type for an algorithm all these operations must be counted. Each basic operation takes a fixed amount of time known for each computer so that the counts can be transfomed to time which is a single measure.

1. *INTRODUCTION*

To solve mathematical problems, a bewildering assortment of algorithms exist. The proliferation of these algorithms is bound to cause some confusion to the user (in any area), who is faced with a given technical problem and would like to know what kind of algorithm should be selected to solve the problem under consideration efficiently.

There seems to be a tendency to choose from available algorithms that are easy to use and well known, (especially amongst occassional users). However, the time and/or money savings etc., justifies the comparison of different algorithms and different versions of the same algorithm in a reliable way.

Unfortunately, there is no clear-cut answer to this question, although there has been much work done in the area.

2. *STATE OF THE ART*

A survey [7] of the techniques or approaches to compare such algorithms, from 50 selected papers (1953-1977), revealed only few factors as the most commonly used, in comparing the speed of mathematical programming algorithms.

The purpose of this paper is to examine these factors from a critical point of view to magnify their drawbacks on conclusions and then introduce a new approach which seems to overcome these difficulties.

2.1 *Factors for comparison of algorithm and their associated difficulties.*

(a) *Computational Speed.*

Refers to the time required to obtain the solution of a problem with a specified precision. By [7], 68% of the comparison methods use CPU time or a normalised form of CPU time which is highly machine and programmer dependent. The only situation where the same CPU time was obtained consistently for one code, was when that code was the only one on the computer, [5]. [7] concludes that "there is a potential source of difficulty with using CPU as the sole performance indicator".

(b) *Number of Functional Evaluations.*

Counts the number of times one or more functions must be evaluated to arrive at a specified precision in the solution. Where several functions and/or derivatives are involved, usually one function is considered the principal and evaluation of other functions/derivatives are measured proportionally to arrive at a single number for comparison. Obviously the whole idea is subjective, although it is expressed in number form. No general rule has been set to choose the principal function and to relate the work required to other functions in terms of the principal function.

In addition, auxiliary computations which may need more time and effort than function evaluations, are ignored in this approach. Combining this and CPU time, as some people do, will only cause more confusion and increase the error based on what was said. [7] and [11], both concluded that it is not a relaible measure.

(c) *Number of iterations.*

This measure refers to the number of times, the whole algorithm must be reiterated, to arrive at the specified precision in the solution.

The 1987 COAL Meeting:

The NATO Advanced Research Workshop on

ALGORITHMS AND MODEL FORMULATIONS IN MATHEMATICAL PROGRAMMING

at

Chr. Michelsen Institute, Bergen, Norway

on

June 15-19, 1987

The COAL workshop has been funded as a NATO Advanced Research Workshop under the Double Jump program. This program is used to support meetings that are viewed as being particularly interesting for both industry and research. As a result of the NATO funding, the number of participants will be limited to about 40. (There is already a waiting list.)

The structure of the workshop will be as follows: The morning sessions will be devoted to plenary discussions with introductory keynote speakers. The first day will be devoted to the interaction between modelling and algorithmic developments in mathematical programming in general. We wish to analyze the current situation and discuss where we wish to go in the years to come. The next four days will be used to see this interaction from four different viewpoints, namely linear programming, integer programming and combinatorial optimization, network flows, and stochastic programming. In addition to the key-note speakers, we plan to let some of the participants present their work in the plenary sessions in order to broaden the input to the discussions. The afternoon sessions will be used for presentations by the participants.

Invited speakers are Professor John Mulvey, Princeton; Professor Walter Murray, Stanford University; Professor Alexander Rinnooy Kan, Erasmus University; Professor Roger J-B Wets, University of California at Davis, and Dr. Ellis Johnson, IBM.

Submit enquiries to: Dr. Stein W. Wallace, COAL-87, Chr. Michelsen Institute, N-5036 FANTOFT-Bergen, Norway.

MARTIN BEALE MEMORIAL SYMPOSIUM

A symposium will be held at the Royal Society, London from July 6th-8th, 1987 in memory of Professor E. M. L. Beale, sponsored by Imperial College, the Institute of Mathematics and its Applications, the Institute of Statisticians, the Mathematical Programming Society, the Operational Research Society, the Royal Statistical Society, and Scicon Limited.

Requests for registration forms and further information should be sent to:

Mrs. B. A. Peberdy
Scicon Limited
Wavendon Tower
Wavendon
Milton Keynes MK17 8LX
England

SYMPOSIUM ON PARALLEL OPTIMIZATION

August 10 - 12, 1987

Computer Sciences Dept.
University of Wisconsin
Madison, Wisconsin
USA

Partial List of Invited Speakers

D. P. Bertsekas
Y. Censor
M. D. Chang
G. B. Dantzig
R. DeLeone
R. S. Dembo
J. G. Ecker
L. Grandinetti
S.-P. Han
J. K. Ho
O. L. Mangasarian*
R. R. Meyer*
J. M. Mulvey
J.-S. Pang
A. H. G. Rinnooy Kan
K. Ritter
J. B. Rosen
R. B. Schnabel
D. Sorensen
S. A. Zenios

**Symposium Co-chairmen*

This symposium will address computational aspects of parallel algorithms for optimization. Emphasis will be on algorithms implementable on parallel and vector architectures. Further information may be obtained by writing to the SPO Secretary or to either of the symposium co-chairmen at the above address. The symposium will be supported by grants from the Air Force Office of Scientific Research and from the Office of Naval Research.

A METHODOLOGY FOR ALGORITHM COMPARISON IN MATHEMATICAL PROGRAMMING

by
E. Shayan
Department of Mathematics
and Operations Research
Footscray Institute of Technology

November, 1986

CONTENTS

APPLICATION FOR MEMBERSHIP

Mail to: The Mathematical Programming Society, Inc.
c/o International Statistical Institute
428 Prinses Beatrixlaan, 2270 AZ Voorburg, The Netherlands

Checks or money orders should be made payable to The Mathematical Programming Society, Inc., in one of the currencies listed below.

Dues for 1987, covering subscription to volumes 37-39 of *Mathematical Programming*, are Dfl. 103.50 (or $ 45.00 or DM 95.00 or £ 30.00 or FF 300.00 or Sw.Fr. 75.00).

Subscription to volumes 30-31 of *Mathematical Programming Studies*, available to members only, is Dfl. 50.00 (or $ 22.00 or DM 44.50 or £ 15.00 or FF 145.00 or Sw.Fr. 36.00).

I wish to enroll as a member of the Society. My subscription(s) is (are) for my personal use and not for the benefit of any library or institution. I enclose payments as follows:

Dues for 1987	....
Subscription to Studies 30-31	....
Total	....

Name (please print)	....
Mailing address (please print)	....
	....
	....
Signature	....

Student applications. Dues are one-half the above rates. Subscription to the *Studies* is at full rates. Have a faculty member verify your student status and send application with dues to above address.

Faculty verifying status	....
Institution	....

CALL FOR PAPERS

13th International Symposium on Mathematical Programming

The International Symposium on Mathematical Programming is the triennial scientific meeting of the mathematical Programming Society. The 13th symposium will be held on August 29 - September 2, 1988, at Chuo University, Tokyo, Japan. Participation is open. Those who want to present a paper are invited to submit the title of the paper by March 1, 1988, and the abstract by May 1, 1988. Acceptance of papers will be notified by July 1, 1988. For further information, please contact the Organizing Committee for the 13th International Symposium on Mathematical Programming, c/o The Operations Research Society of Japan, Gakkai Center Bldg., 2-4-16 Yayoi, Bunkyo-ku, Tokyo 113, Japan.

MP INTEGRATED MATHEMATICAL PROGRAMMING PACKAGE

Choose any subset of the following eight systems to create a suitable package

MP1-LPROG	linear programming system
MP2-QPROG	quadratic programming system
MP3-LCPROG	linear complementarity system
MP4-BIMAT	bi-matrix game system
MP5-FIXPT	fixedpoint computing system
MP6-NLPROG	non-linear programming system
MP7-MIPROG	mixed integer programming system
MP8-NLCPROG	non-linear complementarity system

These systems share

* user friendly input/edit segment
* store output on diskette for future use
* convert data files between systems
* parametric versions available
* 8087 support available

Price $80.00 for 1, $160 for 2, $210 for 3, $260 for 4, $290 for 5, $320 for 6, $340 for 7 and $360 for 8 systems. Site license available.

We are introducing a new package of interest to numerical mathematicians and scientists, the Scientific Systems Package SS. At present this package consists of the system

SS1 - CURVEFIT a curve fitting system

and will contain systems for solving differential equations, nonlinear equations and other scientific software. The price of SS1, for a single machine license is $80.00.

For further information please contact:

SCI COMPUTING,
2640 PARKRIDGE DRIVE,
ANN ARBOR, MICHIGAN 48103
313-747-8330
313-995-5976

Notes from the COAL Chairman

New COAL member: On August 30 I was informed that the Executive Committee of the Society approved the nomination of James B. Orlin as a member of COAL. Welcome Jim! We look forward to working with you.

Retiring COAL member : Harlan Crowder indicated that he would like to step down from the Committee. Harland did a lot for COAL in the early years and I am sure he remains very sympathetic to our efforts.

SIGCMP (ACM) : is the process of being formed. Anybody interested in serving as an officer may contact Jim Kreuser or Bob Meyer.

Guidelines : No new information.

COAL NATO ARW in Bergen (Norway) June15 – 19, 1987 Details of the Advanced Research Workshop on algorithms and Model Formulation in Mathematical may be found on page 21.

Mathematical models for Decision Support : Under this title Gautam Mitra organizes an Advanced Study Institute in Val d'Isere (France) from July 26 until August 6, 1987.

Topics covered include: structured modelling, computer assisted modelling, fuzzy sets and systems, logic programming and relational data models. Invited speakers include many old COAL friends (Lootsma, Greenberg, Rijckaert) as well as people from other backgrounds.

Jan Telgen

Notes from the U.S. Co-Editor

This issue of the COAL Newsletter contains two contributed papers, the first by E. Shayan on a methodology for algorithm comparison in mathematical programming, and the second by James Ho and Rangarajan Sundarraj on a generator for LP test problems. This summer we are fortunate to have a number of major meetings in mathematical programming, details of which may be found in this Newsletter. This is the first issue of the Newsletter to contain paid advertising. Suitable advertisements are welcomed for future issues at a rate of $100 for a full page.

Publication of this issue was delayed due to a variety of factors, including my frequent absences from Madison over the past few months. I will endeavor to get future issues distributed in a more timely fashion, and I expect the next issue to be distributed in August.

Robert R. Meyer

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 provide a vehicle for the rapid dissemination of new results in computational mathematical programming. To date, our profession has not developed a clear understanding of 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.

ROBERT R. MEYER
UNIVERSITY OF WISCONSIN-MADISON
COMPUTER SCIENCES DEPARTMENT
1210 WEST DAYTON STREET
MADISON, WISCONSIN 53706

Mathematical Programming Society

Committee on Algorithms Newsletter

NO. 15	JENS CLAUSEN	EDITORS
DECEMBER 1986	ROBERT R. MEYER	

CONTENTS