to collect test problems and subsequently to disseminate them is placed on experimenters, and for many this is an unreasonable burden. Thus we come back to the need for some group or institution to be willing to do this and to advertise the availability of these collections, so that anyone wishing to test a new idea knows where to go to get "good" test problems.

The creation, updating, maintenance, and dissemination of these collections is a grubby, unrewarding, and expensive exercise, especially for large-scale problems, not simply because they are large, but also because they frequently arise in practice, are proprietary, and must be "sanitized" before they are disseminated. This could continue to be left to the individual groups or institutions who, as in the past, collected test problems for their own use and were willing uncomplainingly to shoulder the burden of dissemination, but this seems unfair, and in any case, appears not to work as smoothly as one would like. The recent case of AT&T Bell Labs'testing of the new projective method is an example of this. It was claimed that test problems were too hard to obtain, so they used the only ones they had, wich were proprietary. Perhaps the situation would have been ameliorated had better collections of test problems been available or had the existing ones been better publicized.

In any case, one the side benefits of the computational evaluation (described elsewhere in this newsletter) of the new projective method that the National Bureau of Standards, the Committee on Algorithms, and AT&T Bell Labs are planning to conduct will be a collection of linear programming test problems of a variety of types, sizes, difficulties, and origins. We are arranging for funding to develop, maintain, update, and disseminate this collection, and we hope this will help avoid future such occurrences of exciting performance results whose scientific substantiation is unavailable for public scrutiny. Furthermore,we can all hope this is a sign that funding agencies are now willing to fund more such computational testing and test problem collection efforts.

REFERENCES

(1) Crowder, H.P., Dembo, R.S., and Mulvey, J.M., On reporting computational experiments with mathematical software, ACM TOMS Vol. 5, no. 2, June 1979, pp 193-203.

(2) Greenberg, H.J., ed., Design and implementation of optimization software, NATO Advanced Study Institute Series, Sijthoff & Noordhoff, the Netherlands, 1978.

(3) Hoffman, K.L., Jackson, R.H.F., and Telgen, J., eds., Mathematical Programming Study on Computational Mathematical Programming, to appear.

(4) Mulvey, J.M., ed., Evaluating mathematical programming techniques, Proc. conference held at National Bureau of Standards, Boulder, Co January 5-6, 1981, Springer-Verlag Lecture Notes in Economics and Mathematical Systems No. 199, Springer-Verlag, New York, 1982.

(5) Schittkowski, K.S., ed., Proc. of Nato Advanced Study Institute on Computational Mathematical Programming held in Bad Windsheim, FRG, July 22 - August 3, 1984, to appear.

A NOTE ON THE NUMBER OF SIMPLEX ITERATIONS

Snjólfur Ólafsson, Science Institute, University of Iceland, Dunhaga 3, 107 Reykjavík, Iceland.

P.O. Lindberg, Department of Mathematics, Royal Institute of Technology, 10044 Stockholm, Sweden.

This note is a comment on (1). There Haverly reports on a study that was made as a response to our article (2), where we reported on Simplex iteration counts higher than he expexted. The experimental base for these results is given in fuller detail in (3).

In few words, for a d x d assignment problem (i.e. $\max \sum_{ij}\sum c_{ij} x_{ij} : \sum_{i} x_{ij} = 1$ all j , $\sum_{j} x_{ij} = 1$ all i , $x_{ij} \geq 0$) we observed the iteration count growing as $d^{1.57}$. The number of rows (i.e. independent equality constraints) is 2d-1 and hence the growth of the iteration count is considerably faster than linear in the number of rows. On the other hand, the empirical experience and various studies (e.g. Haverly's) show linear growth (usually with factor between 1 and 5).

The main explanation for this (great) difference is that in our study we choose the worst basis as the starting basis, whereas usually one tries to choose the starting basis in some clever way, or - like Haverly - in some neutral way.

The difference between our observed growth and the common experience thus indicate that one gets better and better starting points, compared to the difference of the best and worst point, as the problem size increases.

Note also that even though the results presented in (1) are in agreement with the linear growth, the study is far too small to choose between the linear and the exponential model.

A secondary explanation is that PNET only searches through a small part of the (nonbasic) variables for the one with lowest reduced cost. More precicely PNET searches through one row (i.e. all variables with a common origin) at each iteration. Compared to the "standard variant", this variant givs higher iteration count, but uses shorter computing time for each iteration, and usually also shorter total solution time.

There are two issues raised here that bear further explanation. One concerns the meaning of "thorough" experimentation, and the other concerns the protection of commercial interests. It is not the purpose of this article to explain in detail how properly to conduct a scientific experiment, or more specifically, a computational mathematical programming experiment. Others (see the references and their references) have done excellent jobs of that before. Suffice it to say that, at a minimum, one needs to develop a sound experimental design, including identification of objectives, influential variables, performance measures, and test problems and that one must publish the results of that experiment in such a way that the experiment may be reproduced.

Requiring public reproducibility, unfortunately, is sometimes perceived as detrimental to corporate proprietary or commercial interests. As Crowder, et al pointed out in the guidelines, full reproducibility of experimental results requires a listing of the program and a detailed description of the inputs. In many cases, that program is proprietary. Their definition of reproducibility therefore (which I like), is that the autors themselves should be able to replicate the experiment. This, however, is a middle-of-the-road approach that tends to leave the scientific community dissatisfied, since full disclosure is desired by them. The situation is exacerbated when a computational experimenter chooses to use propietary test problems in the expiriment and subsequently reports the results in the open literature. In such situations, nothing is disclosed that can be substantiated by other researchers, and what are left are unverifiable claims that we are expected to accept on faith. This, of course, is unacceptable to the scientific community.

The situation can be improved for the scientific literature by insisting that the quidelines given in (1) be followed more rigoroulsy. This would mean among other things that a complete description of the mathematical algorithm be given, even if the code itself cannot be, that non-proprietary test problems be used and that these problems be made available to others who wish to attempt verification, evaluation, or comparison. It follows that the burden

mathematical programming software, but also a heightened awareness of the need for careful substantiation of performance claims. It is rare nowadays to see statements like the following that appeared in the literature in 1968:

"Since the methods were coded for different machines in different languages by different programmers, there is little point in giving a detailed assesment of the results, particulary as so many of the problems were degenerate. However the results show that ....." (Underlining added.)

One reason such statements are rare is that algorithm developers have come to understand that computational testing of new algorithmic ideas is an important and integral part of the development process and that this testing is to be viewed as a process of experimentation not unlike that used in other branches of science. As such, these experiments should be conducted with the same rigor and following the same dictates of the scientific method as required of scientists in these other branches of science. The development of this view of computational experimentation was aided in part by the publication (1) of the quidelines for reporting results of computational experimentation by Harlan Crowder, Ron Dembo, and John Mulvey, who at the time were members of COAL and were operating under a charge from the committee and community to develop these guidelines. Since their publication, a few of the journals of the operations research community have adopted them (or a variant of them) as part of their editorial policy. In untold other cases, editors, associate editors, and referees have privately adopted them to aid in the publication process.

The essential point I wish to make is that research, development, and testing in computational mathematical programming is science, and we all have a responsibility to conduct our research and development efforts as scientists. This means thorough mathematical analysis and thorough computational experimentation. It also means that the published results of this analysis and experimentation effort must be documented in a manner that allows checking or substantiating the reproducibility of the results.

We have made a minor study on the effect of only searching through a given part of the variables. The stochastic Simplex method (which was the research object of (2) and (3)) is comparable to always choosing the first negative reduced cost. We also changed PNET to search through 4 rows at a time, and finally to search through all the variables. We can summarize the results in the following table which gives iteration counts from the worst basis as depending on the set of variables scanned for lowest reduced cost. The model in the next to the last column is based on the numbers in the table, and the last column is taken from (2):

DIMENSION:	20	23	26	45	58	75	model	(2)
Stoch.	288	362	472	1198	1986	3186	$1.25d^{1.81}$	$1.2\ d^{1.83}$
one row	114	130	204	425	569	945	$1.07d^{1.56}$	$1.1\ d^{1.57}$
4 rows	93	118	146	327	481	693	$1.03d^{1.51}$	
all rows	69	93	114	267	389	543	$0.71d^{1.55}$	

We see that there are big differences in the iteration counts for the 4 variants, but still they all fit well to almost the same growth rate. The iteration count 69 in this table can be compared to the iteration counts 38-52 in (1). This shows that for this problem size, then starting from an all slack basis reduces the number of iterations with about 35% as compared to starting in the worst basis.

Anyhow, the point of (2) and (3) was (and is) that the order of growth of the iteration count for the stochastic Simplex method is not so much higher than for the usual Simplex method, giving some insight in why the Simplex method is so good.

(1) C.A. Haverly: "The number of Simplex Iterations for a General LP Code on Network Problems", COAL Newsletter 11 (1984), 16-19.

(2) S. Ólafsson & P.O. Lindberg: "On the Lengths of Simplex Paths" COAL Newsletter 10 (1984), 23-27.

(3) S. Ólafsson & P.O. Lindberg: "On the Lengths of Simplex Paths: the Assignment Case" Mathematical Programming 30 (1984),243-260.

COMPUTATIONAL EXPERIENCE WITH EXTERNAL PIVOTING

H.A. Eiselt and C.-L. Sandblom
Concordia University
Montreal, Quebec, Canada

R. DeMarr
University of New Mexico
Albuquerque, New Mexico, U.S.A.

1. Introduction

For many years, improvements to the simplex methods have been sought. This report is concerned with a new version of the simplex methods, called external pivoting. Its principle was first described by DeMarr [1] and later developed by Eiselt and Sandblom [2]; it was independently found by Murty and Fathi [4]. In the next section we briefly describe some of the basic ideas involved as well as some preliminary testing. In the succeeding section the design of our test series is described and in the last section the results of our test runs are displayed and discussed.

2. The Principle of External Pivoting and First Results

External pivoting is a principle that can easily be incorporated in any of the simplex methods. The version described here is adapted to the primal (two phase) simplex algorithm. At any point during the iterations, some set of pivot eligible columns in the tableau is selected to generate a new column as some nonnegative linear combination. Any column generated in this way will be called an external column as opposed to the internal columns corresponding to the given decision, slack, excess or artificial variables. Bringing a new external column into the basis is called external pivoting in contrast to internal pivoting which is the usual pivoting on internal columns. In an external pivoting step, the values of all variables used to generate the external column are simultaneously and implicitly increased. This resembles block pivoting and other feasible direction methods but avoids some of their drawbacks.

The discussion that day covered many topics (not the least of which was the paperwork required to create such a group within a society that was not organized along these lines), and out of that discussion, and the subsequent correspondence with Mike Powell there developed a set of goals and objectives for what came to be known as the Committee on Algorithms (COAL) of the Mathematical Programming Society. In brief, these objectives are to serve as a focal point for information about mathematical programming software and algorithms, to develop suitable methods for comparing algorithms and software, and to encourage the use of whatever standards and guidelines exist for developing, documenting, and testing new algorithms and software in mathematical programming.

The committee was, and is, not the only group working on these topics. Many other similar efforts were undertaken in other societies and at other institutions, too numerous to mention. Throughout the intervening years, liaisons with these other organizations were maintained in various ways, sometimes by having a representative serve a term on COAL.

Since its creation, the committee has served the Mathematical Programming Society and the community by organizing sessions at various national and international meetings and by organizing entire conferences on computational methematical programming. It has also developed guidelines for the proper conduct of a computational experiment, and for the proper reporting of the results of such an experiment. (References for much of our work are included in the proceedings of the 1977, 1980, and 1984 COAL conferences (2,4,5) and in the MP Study (3).) Another concern of the committee is to develop graded sets of test problems for the different areas of mathematical programming. This need of the community was identified during the first organizational meeting 12 years ago; it has also been one of the most difficult to satisfy. In the end some such sets were developed by various researchers, and the committee simply served to connect those who had problems with those who needed them.
While there is much yet to be done, there has been not only great progress in the area of improving the state-of-the-art of computational mathematical programming and the evaluation of

ON THE STATE-OF-THE-ART OF COMPUTATIONAL TESTING OF MATHEMATICAL PROGRAMMING ALGORITHMS

by

Richard H.F. Jackson

Center for Applied Mathematics
National Bureau of Standards
Gaithersburg, MD 20899

After the special session on the new projective method for linear programming by N. Karmarkar that was held at the Dallas meeting of ORSA/TIMS, I was asked to write a short article summarizing some of the philosophical statements about computational testing I made during the discussion session following Karmarkar's presentation. I could not help but begin by reflecting on the history of computational testing of mathematical programming algorithms and noting that some of the events that transpired recently were predicted a long time ago.

It will be twelve years this summer since John Tomlin, Jerry Kreuser, Michel Balinsky, Larry Haverly and I discussed the creation of a subgroup of the Mathematical Programming Society which would somehow provide information to the mathematical programming community about latest developments in mathematical programming software, and the availability and quality of that software.

We were motivated by the increasing amount of research in computational mathematical programming and the increasing number of published papers containing overlapping and conflicting claims of performance of new algorithms being proposed. We felt that something had to be done to improve the state-of-the-art of computational testing and experimentation in mathematical programming before events got out of hand. We further felt (correctly as it turned out) that before long not just pride of authorship would be at stake in these comparisons, but also commercial interests, as the state-of-the-art of mathematical programming improved pari passu its use and acceptance in business, industry, and government.

One can show (Eiselt and Sandblom [2]) that the optimal solution remains unchanged when adding an external column to the problem since this is equivalent to adding a weakly or strongly redundant constraint to the corresponding dual problem. A word of caution is, however, in order. The successive creation of external columns introduces linear dependency which may cause numerical instability.

Several strategies for external pivoting exist. In particular, at each iteration the user may

(a) decide whether to use internal or external pivoting;
(b) specify the set of columns for generating the new external column (in case external pivoting is chosen);
(c) specify the weights associated with the generating colums (again if external pivoting is to be performed).

In a preliminary test involving thirty randomly generated problems (for details see Eiselt and Sandblom [2]), (a) and (b) above were fixed and only the selection of weights in (c) was varied. Pure external pivoting was used, i.e. an external column was generated in each iteration and as generating columns all pivot eligible columns were selected. The following weights were specified: M_1, equal weights; M_2, weights equal to the current reduced costs; and M_3, weights equal to the objective function value increase if the corresponding column were selected as the pivot column. Method M_3 came out a clear winner in terms of iterations, although many pivots had to be determined in each iteration. All three methods quickly achieved relatively high objective function values as opposed to M_0, the standard simplex method (with steepest unit ascent pivot column selection criterion) which was used as a benchmark. But after this good initial performance, all three "pure" external pivoting methods were slow in reaching optimality. As expected, numerical instability was also present. Thus other methods were designed to overcome these difficulties: with M_4, external pivoting is performed once

initially, followed by the regular primal simplex method. M_5 uses external pivoting twice before returning to the standard method and M_6 uses either internal or external pivoting, according to the larger objective function value increase. Again, thirty test problems were solved with M_4 as the clear winner. Considering only the larger problems in the test series, M_4 needed about twenty percent less iterations that M_0. These findings were confirmed when another forty test problems were solved. This performance encouraged us to test the method M_4 further in a series of larger problems.

3. The Test Series

The test problems were generated as follows: the number of variables n was spaced evenly between 75 and 140 and the number of constraints m was randomly generated between $\frac{1}{2}$n and 2n. The density of the problems was between 2/n and 1/4 before adding slacks. The values of all nonzero technological coefficients a_{ij} as well as the objective function coefficients c_j were positive integers between 1 and 20. Their respective positions in the tableau were also generated at random. The right-hand side values b_i were determined as $b_i = t_i^{-1/2} \Sigma\, a_{ij}\, r_j$ where t_i is the number of nonzero coefficients in the i-th row and r_j are single-digit random numbers. This relation was chosen in order to avoid too large a degree of redundancy while retaining a certain degree of randomness. Zero rows and/or columns were deleted. Note that our problems are sparse, as opposed to those of, say, Sherali, Soyster and Baines [5] who use the principles of Kuhn and Quandt [3] to generate their (dense) problems.

After generating each problem, it was solved by the Multi Purpose Optimization System (MPOS) package on a CDC CYBER 835 computer using the revised simplex method without scaling. The package uses the steepest unit ascent criterion, i.e. our benchmark method M_0. Then the same problem is regenerated and solved using

6. What test problems should be used?
NBS and COAL will take the responsibility for collecting the test set which we envision will include small-, medium-, and large-scale problems chosen from among hand-generated and real-world examples. Various sparsity structures will be represented and the set will include degenerate, unbounded and infeasible examples. We propose that the actual tests be selected from among this collection.

7. What computational results should be reported?
We recommend that the code be instrumented, either automatically, by hand, or in some combination, to produce iteration and/or operation counts and time for each of the three phases -preprocessing, solving, and post- optimality analysis. For the solution phase, we should definitely obtain a count of the number of iterations and the approximate amount of work per iteration. Some Fortran systems include options to obtain various run-time statistics including execution frequencies, time spent in designated loops, etc. Thus, several runs, using different versions or different system options, may have to be made to achieve this. Reported time should optimally be obtained from runs on a dedicated machine.

In our opinion, the results of such an extensive test would provide the community with a solid basis for assessing the new algorithm.

(1) H. Crowder, R. Dembo and J. Mulvey, "On Reporting Computational Experiments with Mathematical Software", ACM Transactions on Mathematical Software, Vol. 5, No. 2, 1979, pp. 193-203.

2. What format should the test problems be in?
All problems will be provided in MPS format which is the de facto industry standard.

3. What about preprocessing?
Most LP codes have a preferred formulation for the problems they solve; their performance on mathematically equivalent formulations can be radically different. For example, by representing the special structures of simple bounded variables, range constraints, and special ordered sets independently of the constraint matrix, modern simplex codes significantly reduce the number of operations needed to solve the problem. In addition, these codes typically have built-in preprocessors which transform the problems into efficient and numerically stable forms. Karmarkar has stated on several occasions that he has hand transformed problems into the preferred form for his algorithm. In order to conduct a fair test of Karmarkar's code, this preprocessing phase must be allowed. If an LP is transformed, however, then it must be done according to an automatic procedure and that automatic preprocessing procedure must be specified consistent with issue (1) above.

4. What about the issue of convergence?
Karmarkar's algorithm obtains the answer to a theoretical, specified precision. Here we recommend running Karmarkar's code for a range of accuracies, e.g. 10**(-3), 10**(-5), 10**(-7), and 10**(-9), in order to understand better the overall performance of the algorithm. All of the problems in the test set will have solutions known to accuracies greater than 10**(-9).

5. What about post-processing?
Most simplex codes provide some degree of post-optimality analysis. Minimally, they supply the Langrange multipliers (also known as dual variables). The question then becomes whether Karmarkar's code should be required to compute the multipliers as part of its job. Our recommendation is that it should since users of LP almost always want this information. Again, the procedure for computing the multipliers from Karmarkar's solution should be specified.

method M_4. Since MPOS is a proprietary code, the first iteration (with external pivoting) had to be done by our own simplex code before the problem was read into MPOS.

4. The Results

The performance of the two methods M_0 and M_4 was judged based on the number of iterations required to reach optimality. For the sixty test problems we found that, on average, M_4 required only 80.12 percent as many iterations as M_0. The superiority of M_4 over M_0 did not appear to be related to sparsity or to the ratio m/n; nor did the convergence pattern of the two methods differ in any noticeable way. Interestingly enough, it turned out that the more iterations required for M_0, the more pronounced was the advantage of M_4. Specifically, for the twelve test problems that required more than 350 iterations to reach optimality with M_0, the M_4 method needed only slightly more than half of M_0's iterations. For details, see the table below, where $it(M_0)$ and $it(M_4)$ denotes the number of iterations required by M_0 and M_4, respectively.

$it(M_0)$	Number of problems	Average $100it(M_4)/it(M_0)$
0 - 99	6	126.71
100 - 149	10	94.86
150 - 199	10	80.32
200 - 249	10	76.09
250 - 349	12	72.12
350 - 650	12	55.75
Total:	60	80.12

In order to further improve on these results, other external pivoting strategies are currently being designed and tested.

References

[1] R. DeMarr, "External Pivoting in Linear Programming", Abstracts of the American Mathematical Society, issue 22, vol. 4, no. 1 (1983)

[2] H.A. Eiselt and C.-L. Sandblom, "External Pivoting in the Primal Simplex Algorithm", Revised version of working paper no. 84-012, Faculty of Commerce and Administration, Concordia University, Montreal, Canada (1984)

[3] H.W. Kuhn and R.E. Quandt, "An Experimental Study of the Simplex Method", Proceedings of the Symposia in Applied Mathematics XV, American Mathematical Society, Providence, R.I. (1963), 103 - 124

[4] K.G. Murty and Y. Fathi, "A Feasible Direction Method for Linear Programming", Operations Research Letters 3 (1984), 121 - 127

[5] H.D. Sherali, A.L. Soyster and S.G. Baines, "Nonadjacent Extreme Point Methods for Solving Linear Programs", Naval Research Logistics Quarterly 30 (1983), 145 - 161

PROPOSAL FOR TESTING KARMARKAR'S ALGORITHM

Paul Boggs, Karla Hoffman, Ric Jackson

A new and interesting algorithm for linear programming (LP) has recently been proposed by N. Karmarkar of AT&T Bell Labs. It has some theoretically provocative properties which suggest that it could be a serious competitor to the simplex algorithm. In addition, some early promising computational results have been reported by Karmarkar, but many researchers and users want to see the results of a scientifically planned, well-documented test effort.

The National Bureau of Standards (NBS) and the Committee on Algorithms (COAL) of the Mathematical Programming Society, in cooperation with AT&T Bell Labs, propose to provide the first set of such tests. Our primary goal is to shed light on the performance of Karmarkar's algorithm, thereby exhibiting the strengths and weaknesses of the procedure. In particular, are there classes of problems for which the algorithm is especially effective? Are there aspects of the procedure which need further research? An important second goal of this project is to collect a basic set of LP problems to enhance further testing of this and other LP algorithms.

Most of the general issues involved in the nummerical testing of mathematical software have been set forth in (1). The particular issues relevant to the proposed testing are discussed below.

Issues

1. What algorithm is being tested?

It is important that we have a complete description of "the" algorithm being tested. This does not imply that it is necessary to publish the code, but it does mean (consistent with (1)) that enough information be given so that another version of the code, produced from that description, would perform similarly. The testing group must have enough access to the code being tested so that they can agree that the code does indeed implement the algorithm as described. Since the code may have options appropriate to different problem structures, the values of all input parameters must be specified.

Paul Boggs
Scientific Computing Division
Building 225, Room A151
National Bureau of Standards
Gaithersburg, MD 20899
301/921-3395

Karla Hoffman, Chairman, COAL
Operations Research Division
Building 101, Room A415
National Bureau of Standards
Gaithersburg, MD 20899
301/921-3855

Ric Jackson
Operations Research Division
Building 101, Room A415
National Bureau of Standards
Gaithersburg, MD 20899
301/921-3855

ANNOUNCEMENTS

LINEAR....INTEGER....SEPARABLE....PROGRAMMING TEST PROBLEMS

We wish to test our mathematical programming system over a wide range of test problems and would be very pleased to receive such test data We would like these problems to be supplied on magnetic tapes or floppy diskettes, and ideally in either MPSX or APEX III format. We would be happy to pay for the preparation and handling cost of tapes, diskettes, etc. Suppliers confidentiality will be fully respected if this is required. We ourselves have a range of 14 test problems.

Smallest	50 rows	-	50 columns	-	300 nonzero
Largest	1350 rows	-	1600 columns	-	14000 nonzero

Out of these 6 are integer programming problems.

We would be quite pleased to exchange these test problems.

Please write to Dr. G. Mitra, Department of Mathematics and Statistics, Brunel University, Uxbridge, Middlesex, UB8 3PH or thelephone Uxbridge (0895) 37188, Extension 259, or Mrs. P. Denham on Extension 276.

WORKSHOP ON GLOBAL OPTIMIZATION

A Workshop on Global Optimization will be organized by the International Institute for Applied Systems Analysis (IIASA). It is scheduled to take place in Sopron (Hungary) in the first half of December 1985. Those interested in contributing should contact Alexander H.G. Rinnooy Kan, Econometric Institute, P.O. Box 1738, 3000 DR Rotterdam, The Netherlands, ((10) 525511 ext. 3030)

NUMERICAL TECHNIQUES FOR STOCHASTIC OPTIMIZATION PROBLEMS

The Adaptation and Optimization (ADO) project of the Systems and Decision Sciences program at the International Institute for Applied Systems Analysis is preparing a survey of the techniques currently employed to solve various problems in stochastic linear and nonlinear programming titled "Numerical Techniques for Stochastic Optimization Problems," edited by Yu. Ermoliev and R. Wets. In addition, the project has assembled a collection of experimental computer codes that implement several of the algorithms discussed in the survey. This collection will be made available on computer tape to researchers early in 1985. There is a high degree of variation in the quality of these codes; some are near production-level quality whereas others have been built to test the potential of a proposed method or to solve a specific class of problems. In certain cases it is assumed that the distribution of the random variables (which model the uncertainty) is discrete or has been discretized. In other cases no assumptions are made concerning the distribution - the code itself performs the discretization or simply avoids multidimensional integration.

Ten authors have contributed nine programs to the ADO collection. These programs address a number of different problems and use a variety of techniques to solve them, as indicated in the list below.

Author and Institution	Problem	Method
John Birge, University of Michigan	Multi-stage stochastic linear programs	Nested decomposition for stochastic programming (based on Van Slyke and Wets' L-shaped method)
A. Boehme and Kurt Marti, Hochschule der Bundeswehr Muenchen	general stochastic programs with recourse	Marti's descent stochastic quasigradient method

TESTING KARMARKAR'S ALGORITHM BY COAL/NBS

ARPANET Announcement (January, 1985)

A new and interesting algorithm for linear programming (LP) has recently been proposed by Dr. N. Karmarkar of AT&T Bell Laboratories. It has some theoretically provocative properties which suggest that it could be a serious competitor to the simplex algorithm. In addition, some early promising computational results have been reported by Karmarkar, but many researchers and users want to see the results of a scientifically planned, well-documented test effort.

The National Bureau of Standards (NBS) and the Committee on Algorithms (COAL) of the Mathematical Programming Society, in cooperation with AT&T Bell Laboratories, will provide the first set of such tests. Our primary goal is to shed light on the performance of Karmarkar's algorithm, thereby exhibiting the strenghts and weaknesses of the procedure. An important second goal of this project is to collect a basic set of LP problems to enhance further testing of this and other LP algorithms.

We have recently discussed this test effort with Dr. A. Fronistas and Dr. R. Graham of AT&T Bell Labs and they have agreed to cooperate. The details are currently being worked out.

Many people in the optimization community have already volunteered to submit examples of LP problems to this collection; we would be pleased to hear from any others who have test problems.
Since COAL sees this collection as becoming the basis for an expanded set of LP test problems, we need to consider carefully the details of the organization of the set. Since MPS format is the de facto industry standard, we will probably opt to use it for the test set.

We would appreciate your thougts on any aspect of this endeavor. Unfortunately, we cannot be conveniently contacted over the ARPANET. We can be contacted throught the time-honored medium of the US Mail at the address below or by phone at the numbers listed.

J.A. Tomlin: "An Experimental approach to Karmarkar's linear programming algorithm", Ketron, Inc., Mountain View, California, November 1984.

M.J. Todd and B.P. Burnell: "An extension of Karmarkar's algorithm for linear programming using dual variables", Technical report no 648, SORIE, Cornell University, January 1985.

Alexei Gaivoronsky V. Glushkov, Institute of Cybernetics, Kiev and IIASA	general stochastic programs with simple constraints	Ermoliev and Gaivoronsky's variants of stochastic quasigradients
Alan King, University of Washington and IIASA	stochastic quadratic programs with simple recourse	Rockafellar and Wets' Lagrangian finite generation method
Larry Nazareth, Computative Decision Support Systems (Berkeley) and IIASA	stochastic linear programs with simple recourse	Nazareth and Wets' inner linearization method
Liqun Qi, University of Wisconsin	stochastic transportation problems with simple recourse.	Qi's forest iteration method
Andrzej Ruszczynski, Institute of Automatic Control, Technical University of Warsaw	general stochastic programs with recourse	Ruszczynski and Syski's method of aggregate stochastic subgradients
Tamas Szantai Technical University of Budapest	stochastic programs with probabilistic constraints	Veinott's supporting hyperplane algorithm
Stein W. Wallace Chr. Michelsen Institute Bergen, Norway	stochastic linear programs with pure network recourse	Van Slyke and Wets' L-shaped algorithm (adapted; uses Schur complementents in bunching)

The ADO tape contains the source code provided by each author and a User's Manual for each program. There are sample input and output files for most of the codes as well. The tape also includes a paper outlining a proposal for a standard input format for routines that solve stochastic programs with recourse and a library of utilities to facilitate the use of the standard format. The tape's contents total some 4.5 million bytes.

The codes in the collection are written in various dialects of FORTRAN and some use subroutines from the IMSL, MINOS, NAG, or QPSOL libraries. The User's Manuals and other papers provided on the tape are intented for use with the UNIX text processor, trofl, although they can be read as-is. Please note that the majority of the codes in the collection are still under development and that no warranty is granted, either expressed or implied.

The ADO collection may be obtained from

Project Secretary
ADO/SDS
IIASA
A-2361 Laxenburg
Austria.

Persons who would like a copy of the collection should send a blank reel of 9-track computer tape to the above address and should include a note indicating their preferences for density and character set (IIASA's computer can generate unlabelled tapes at 1600 or 800 bits per inch using either the ASCII or EBCDIC character codes)

EDITORIAL COLUMN: KARMARKAR'S ALGORITHM
Recent Developments

There have been a number of interesting recent developments related to Karmarkar's algorithm. A formal proposal has been made by a group at NBS (Paul Boggs, Karla Hoffman, Ric Jackson) for rigorous testing with public disclosure of the results. Negotiations between this group and AT&T Bell Laboratories have been in progress since the November ORSA/TIMS meeting in Dallas, and there is optimism that final agreement may be reached in March or April. Following this article there is a copy of an announcement that was distributed via ARPANET. This announces the proposed test and calls for the submission of suitable linear programming test problems. Ric Jackson reports that there has been an excellent response to this call, and many industry sources that were previously reluctant to release proprietary problems have agreed to submit problems that have been modified to protect confidential data. Thus, a major contribution of this test proposal will be the creation of a publically-available collection of real-world, large-scale LP's. A copy of the test proposal is also included in this issue of the COAL Newsletter. There is some hope that results of this test may be available in time for the August meeting of the Mathematical Programming Society in Boston.

On a related note, a correction appears to be in order with regard to an item on Karmarkar's method in the previous COAL Newsletter, where it was stated that "savings of a factor of 50 in comparison to a commercial LP code are reported on one of George Dantzigs energy problems". Professor Richard Cottle of Stanford has been in contact with Dr. Karmarkar on this point, and reports that, as of December, 1984, no runs had been made by Karmarkar on those energy problems. The test problems used by Karmarkar appear to have originated within AT&T.

Recent technical reports on Karmarkar's method that may be of interest to COAL members are:

A. Charnes, T. Song, and M. Wolfe: "An explicit solution sequence and convergence of Karmarkar's algorithm", Research Report CCS 501, Center for Cybernetic Studies, University of Texas, Austin, Texas, November 1984.

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.

Dr. JAN TELGEN
RABOBANK NEDERLAND
LAAN VAN EIKENSTEIN 9
3705 AR ZEIST
THE NETHERLANDS

Mr. T. Steihaug,
STATOIL,
Forus, P. O. Box 300,
STAVANGER, 4001,
Norway.

Mathematical Programming Society

Committee on Algorithms Newsletter

No. 12

May 1985

JAN TELGEN

EDITORS

ROBERT.R. MEYER

Contents: