DISCUSSION PAPER NO. 231

DISSEMINATION AND MAINTENANCE OF MATHEMATICAL PROGRAMMING SOFTWARE: EXPERIENCE WITH M.P.O.S.

Claude Cohen*†
Jack Stein*

August 1976

†Graduate School of Management
*Vogelback Computing Center

Northwestern University, Evanston, IL 60201

Invited paper presented at the IX International Symposium on Mathematical Programming, 23-27 August 1976, Budapest, Hungary.

Abstract

In the past two years, Northwestern University's Computing Center has distributed to a large number of universities and research institutions a mathematical programming package designed for CDC 6000/CYBER computers. MPOS (Multi-Purpose Optimization System) is a simple-to-use integrated Fortran system for solving small to medium size problems in linear, quadratic, and integer programming, by choosing among eleven algorithms. The purpose of this paper is to describe our procedures for distributing and maintaining MPOS.

Our distribution policy is aimed mainly, but not exclusively, to university computer centers. Because of this orientation, a great deal of effort is spent on producing good user documentation. The maintenance activities of MPOS involve several individuals who spend a portion of their time on that project. The modular design of MPOS allows each programmer to be responsible for his/her algorithm. An extensive archive of test problems (supplied by users or machine generated) is used to certify improved or new algorithms. Surveys of the user community at Northwestern and other sites have been useful in evaluating MPOS, determining user needs, and planning the development of future versions.

Table of Contents

ACKNOWLEDGEMENT

The success of MPOS has been made possible by a great team effort. We owe a great deal to Michael Reagan, Robert Grierson, and Irma Waye for their active participation in this project. We also wish to thank Joseph Yozallinas for his work on the earlier design stages.

1. INTRODUCTION

At the previous Symposium on Mathematical Programming in Stanford, California, we presented the design of an optimization system for CDC 6000/CYBER computers [2]. MPOS (Multi Purpose Optimization System) was conceived and implemented for Northwestern University's users in search of a simple and versatile package to solve a variety of optimization problems in linear, quadratic, and integer programming. Distribution of MPOS to other CDC installations started in the summer of 1974; to date, there are over 35 CDC sites worldwide. These are mostly university computer centers but they also include research and government institutions. The purpose of this paper is to trace our MPOS software development efforts from Version 1 to Version 3 and evaluate these efforts from the point of view of maintenance and distribution. We also report on users' reactions and assess future needs and development.

It is worthwhile to recall for what reasons MPOS was developed:

1. In contrast to statistical packages, such as BMDP, SPSS, etc., comprehensive optimization systems were generally unavailable.
2. The teaching of elementary optimization techniques to several hundreds students in management, engineering, and economics required an economical and simple to learn and use system.

In addition, there was the commitment on the part of Vogelback Computing Center to fulfill users' needs and to continue a tradition of high quality software development and distribution efforts.

2. STAGES OF DEVELOPMENT: VERSION 1 TO VERSION 3

MPOS was specifically designed to teach beginners in mathematical programming to utilize the computer; however, a small but growing number of experienced users and researchers are modeling or solving larger and more complex problems.

With its easily learned control language, the user can in a few minutes set up an MPOS problem. All that is required is the ability to write a set of algebraic equations and to specify in English-like commands the functions to be performed. Contrast, for example, the ways in which the following mixed integer program is run on MPOS and on CDC's large scale mathematical programming system APEX [1]; see pages 3 and 4.

$$\text{Maximize } 77.9x_1+76.8x_2+89.6x_3+97.1y_1+31.3y_2$$

$$\text{Subject to } 10.9x_1+3.6x_2-40.8x_3+43.9y_1+7.1y_2 = 82.3$$

$$-86.8x_1+32.7x_2+24.3x_3+13.8y_1-12.6y_2 \leq 77.3$$

$$60.9x_1+68.9x_2+69.0x_3-56.9y_1+22.5y_2 \geq 86.5$$

$$x,y \geq 0$$

$$x_1 \leq 2 \text{ and integral}$$

$$x_2 \leq 3 \text{ and integral}$$

$$x_3 \leq 6 \text{ and integral}$$

$$y_1 \leq 7.0$$

$$y_2 \leq 2.0$$

It should be clear now that the MPOS system and language require little effort on the user's part to translate from the problem statement to computer instructions. Our program library statistics show that MPOS is the second most widely used applications package at Northwestern (after SPSS).

APEX Input:

```
FILE      PROB                                        APEX Control Program
          TITLE   BBMIP TEST PROBLEM
          SET     R.DTLSW=ON,S.VLGSW=ON
          CONVERT SOURCE=DATA,IDENT=MFILE     Set up appropriate APEX files
          READY   SOURCE=MFILE,IDENT=WFILE,BND=CONBND  Define bound set on
          SETUP   SOURCE=WFILE                          continuous variables
          SWITCH  RFDEV=STOP,RFBCH=CONTI
          SWITCH  INTS=INTP                 Interrupts to obtain cont. soln
          SET     W.OBJ=COST,W.RHS=B,R.FBCH=1,R.FDEV=200,W.SCALE=-1.0
          MIXINT          Define sense of optimization and call algorithm
          STATUS
          RECORD  PRINT                         Print final solution
          EXIT
CONTI     RECORD  PRINT
          SET     R.FBCH=200                    Print continuous solution
          CURRENT
INTP      RECORD  PRINT                         Print integer solutions
          CURRENT
STOP      EXIT
ENDFILE
FILE      DATA
LGL       COST(F)
LGL       ROW1(Z)
LGL       ROW2(P)
LGL       ROW3(M)
STR       X1(I),LIMIT=2                 Bounds on integer variables
STR       X2(I),LIMIT=3
STR       X3(I),LIMIT=6                 Problem data
AIJ       COST,X1=77.9
AIJ       ,X2=76.8
AIJ       ,X3=89.6
AIJ       ,Y1=97.1
AIJ       ,Y2=31.3
AIJ       ROW1,X1=10.9
AIJ       ,X2=3.6
AIJ       ,X3=-40.8
AIJ       ,Y1=43.9
AIJ       ,Y2=7.1
AIJ       ROW2,X1=-86.8
AIJ       ,X2=32.7
AIJ       ,X3=24.3
AIJ       Y1=13.8
AIJ       ,Y2=-12.6
AIJ       ROW3,X1=60.9
AIJ       ,X2=68.9
AIJ       ,X3=69
AIJ       ,Y1=-56.9
AIJ       ,Y2=22.5
RHS       ROW1,B=82.3
RHS       ROW2,B=77.3
RHS       ROW3,B=86.5
BND       CONBND,Y1=7                   Bounds on continuous variables
BND       CONBND,Y2=2
ENDFILE
```

MPOS Input:

```
BBMIP                                                  Select Branch& Bound Algorithm
TITLE
----- MIXED INTEGER TEST PROBLEM (BBMIP) -----        Title to appear
INTEGER                                                Declare integer variables
          X1 X2 X3
VARIABLES      (NOT INTEGER)
                    Y1 Y2
MAXIMIZE                                               Specify objective function
         77.9X1 + 76.8X2 + 89.6X3 + 97.1Y1 + 31.3Y2
CONSTRAINTS                                            Specify constraints
          -86.8X1 + 32.7X2 + 24.3X3 + 13.8Y1 - 12.6Y2     < 77.3
          60.9X1 + 68.9X2 + 69.0X3 - 56.9Y1 + 22.5Y2    >   86.5
          10.9X1 + 3.6X2 - 40.8X3 + 43.9Y1 + 7.1Y2       =  82.3
BOUNDS                                                 Set bounds on variables
                    X1   < 2
                    X2   < 3
                    X3   < 6
* THE Y'S ARE CONTINUOUS VARIABLES                     Comment cards have a '*' in Col.
                    Y1   < 7
                    Y2   < 2
* PRINT EVERY ITERATION
PRINT 1                                                Print every tableau
OPTIMIZE                                               Initiate solution
STOP
```

The continuing development of MPOS has been justified by the following needs:

- new or improved algorithms,
- enhanced control language, and
- increasing problem sizes

1. Algorithms

As we had stated in our previous paper [1], our optimization software library evolved from a collection of stand-alone programs to an integrated system. The modular (overlay) design of MPOS facilitates the inclusion of new algorithms. Table 1 shows the variety of algorithms made available in each version.

Table 1

Optimization Algorithms

Versions	Linear	Quadratic	Integer	Other
1	REGULAR REVISED DUAL MINIT	WOLFE BEALE	BBMIP DSZ1IP	
2		LEMKE	GOMORY	APEX-interface
3	PREVISED	SYMQUAD		

REGULAR 2-phase standard tableau simplex

REVISED 2-phase revised simplex

DUAL dual simplex

MINIT primal-dual

WOLFE Wolfe's quadratic

BEALE Beale's quadratic

BBMIP	Branch and Bound Mixed Integer Programming
DSZ1IP	Direct search 0-1 Integer Programming
LEMKE	Lemke's complementary pivot
GOMORY	Gomory's all integer cutting plane
APEX	Interface to CDC's APEX systems
PREVISED	Packed basis revised simplex
SYMQUAD	Van de Panne and Whinston's symmetric quadratic simplex

There are certainly overlaps in each category. For example, why include (or keep in subsequent versions) Wolfe's quadratic algorithm? We believe that it is easier in some classes to teach Wolfe's (short form) algorithm after the regular simplex rather than Lemke's or Beale's. It is also important - for pedagogical purposes - to provide a perspective of several computational techniques to students interested in implementation and the relative performance of algorithms. The disadvantage of keeping "inferior" or duplicate algorithms is in continuing maintenance and documentation activities.

2. Control Language

The features and error detection of the language processor have also been enhanced in response to users' experiences.

We discovered that there are users who mistakenly specify the same variable more than once in a constraint or objective function, i.e., 5X1+3X2+7X1. As a consequence of this, the tableau formed contained inappropriate coefficients. It is, of course, impossible for the language processor to determine what the user intended but detecting this error was relatively easy to implement. On the other hand, there are users who specify 7(X1+X3) expecting MPOS to expand it into 7X1+7X3. Here the intent of the user is clear but parsing such an expression would involve a major revision of the language processor. We felt the programming effort would not be justified, but a satisfactory solution would be to modify the processor to detect such an expression as an error and to emphasize this in the documentation.

One of the most useful commands added in Version 2 was QCHECK to specify if the quadratic function of a QP problem is to be checked for positive definiteness (in the case of minimization) prior to computing the solution. QCHECK GØ will compute and print the smallest eigenvalue and continue on with the chosen QP algorithm, whereas QCHECK STØP terminates if the sign of the eigenvalue violates the sense of optimization.

The inclusion of NOSCALE is also interesting to point out. Since all LP algorithms automatically scale objective function/constraints for greater precision in computations, the intermediate tableaus printed out were also scaled. Instructors of LP informed us that the comparison of textbook tableaus to computer tableaus was important and that students would be confused by these differences. Instead of unscaling every intermediate tableau, we preferred to implement the option of not scaling the data and document the possible loss of accuracy that could result. Table 2 shows a complete list of MPOS command and control phrases.

3. Problem Size

Although MPOS was designed to solve small scale problems (approximately a few hundreds constraints, depending on the algorithm), there has been a continuing effort to make each algorithm solve bigger problems. Upper and lower bounded variables can now be handled implicitly in each optimization category. The PREVISED algorithm stores the basis inverse in a sparse form and SYMQUAD stores the symmetric quadratic objective function in a sparse form. And to make the transition to CDC's APEX system as easy as possible, the MPOS-APEX interface allows the creation of APEX/MPS-compatible data files.

TITLE	Any alphanumeric title
Algorithm name	REGULAR/REVISED/PREVISED/DUAL/MINIT/WOLFE/BEALE/LEMKE/SYMQUAD/BBMIP/DSZ1IP/GOMORY/APEX1/APEX2
INTEGER	List of integer variables
VARIABLES	List of variable names
MAXIMIZE MINIMIZE	Objective function definition
CONSTRAINTS	Constraints definition
CONSTRAINTS m	Constraints definition for matrix, tableau, or packed input
BOUNDS	Individual variable lower/upper bound
BNDALL value	Upper bound on all variables
BNDINT value	Upper bound on all integer variables
BNDOBJ value	Objective function upper-bound (BBMIP and DSO1IP)
EPSILON value	Perturbation factor
TOLERANCE value	Accuracy check
LIMIT n	Maximum number of iterations
RSCALE k	Row scaling factor (DSZ1IP and GOMORY)
NOSCALE	Do not scale LP data
PRINT	Print initial and final tableaus
PRINT i	Print every i-th tableau
CHECK	Numerical check on LP or QP final solution
QCHECK STOP	Check positive definiteness of Q matrix in QP and abort the problem if the check fails.
QCHECK GO	Check positive definiteness of Q matrix in QP and solve the problem even if the check fails.
RNGOBJ	Range on objective function coefficients
RNGRHS	Range on right-hand side coefficients
REWIND	Rewind the file DATFIL
GETFILE	Retrieve data file
READ DATFIL	Read alternate input file
READ READ INPUT	Read alternate input from INPUT file
PACKED	Packed input
TABLEAU	Tableau input
MATRIX	Matrix input
FORMAT	FORTRAN variable format
OBJROW i	Select row i as objective function
RHSCOL j	Select column j as right hand side
MAXCM f1	Set maximum central memory field length (octal)
MAXECS f1	Set maximum ECS field length (octal)
OPTIMIZE or END	Initiate problem solution
ENDAPEX	Initiate APEX file creation
STOP	Stop run
*	(any comment card)

Organization of maintenance and distribution

The maintenance of MPOS involves five persons who spend a portion of their time on that project (we estimate the total to be 1.75 person-years). At present, the system consists of over 100 subroutines and 23,000 lines of code. The maintenance task is made easier by "decentralized"programming activities: each module (language processor/operating system interface or computational algorithm) is the responsibility of one individual. Maintenance consists of correcting bugs in existing versions, coding modified or newer algorithms and testing them on our extensive collection of test problems. The latter process is partially automated through the use of a utility program that compares output files.

Proper maintenance is subject to the responsiveness of both developers and users. The developers must carefully test each new version prior to releasing it to our users and to other CDC sites (which often have different operating systems). Users also have the responsibility to report problems or errors encountered. To facilitate this communication, we provide in our User's Guide [2] an Error Report Form, and we also are available for telephone consultation. When a bug is reported, it is immediately investigated and fixed as soon as possible. The modifications required are mailed, usually in the form of a card deck, to all sites which have a maintenance contract.

To assess the installation of the system and the reactions of users at large, a survey questionnaire was mailed to MPOS liaison persons at all sites. Sixty five percent reported back valuable information and suggestions. One site (University of Manchester) supplied us with a CDC 7600 version of MPOS. That version is accessed by more than a dozen universities linked to Manchester's regional computer network system.

MPOS is distributed for a small fee with source code and several test problems provided. A one year free maintenance is provided with the initial order. At the end of each year, the site has the option to renew its maintenance agreement at a cost of one half the current purchase price. This agreement entitles the site to consultation, correction of bugs, and new versions. Currently, we have produced one version per year and have released an average of two correction/updates per year. Finally, revisions of the user's documentation accompany each new version. Computational procedures are briefly sketched in the manual and numerous sample problems with output listings are included for each algorithm.

3. EVALUATION AND CONCLUSIONS

MPOS has been a success, as evidenced by its extensive usage at Northwestern and by the number of sites which ordered it, because it fills a need. While originally conceived as a teaching aid in management/operations research courses, researchers found that they could get faster throughput of their problems with MPOS than was possible with complicated and "obscure" systems like APEX. In addition, the use of a local MPOS/ONLINE version through a text editor makes it convenient to solve classroom or other problems from a terminal.

The design and implementation of MPOS have presented numerous interesting and challenging problems. Among these are:

- compatibility with a wide range of CDC operating systems (SCOPE 3.3, SCOPE 3.4, KRONOS, NOS)
- coding of the packed revised simplex
- flexibility for storing branch and bound trees in Central Memory or Extended Core Storage.

We also realize that there are many needs MPOS does not fulfill. Among these are the solution of very large problems (e.g., GUB constraints) and a truly interactive version that allows one to control problem solution and interpose decisions. We hope future versions will meet these needs. Currently, MPOS is only available for CDC 6000/CYBER computers and is not portable. Plans are underway to convert MPOS to other systems.

REFERENCES

1. APEX-I/II Reference Manuals, Control Data Corp., Publication No. 5915800C/59158100, 1974.

2. Cohen, C., M. Reagan, J. Stein, and J. Yozallinas, "Design of an optimization system for university use", presented at the VIII Math. Prog. Symposium, August 1973, Stanford, California, available as Discussion Paper No. 48, Center for Math. Studies in Economics and Management Science, Northwestern University.

3. Cohen, C., J. Stein, MPOS - Multi Purpose Optimization System Version 3 User's Guide, July 1976, Vogelback Computing Center, Northwestern University.