DISCUSSION PAPER NO. 48

DESIGN OF AN OPTIMIZATION SYSTEM FOR UNIVERSITY USE

Claude Cohen*†
Michael D. Reagan*
Jack R. Stein*

Revised April 1974

*Vogelback Computing Center
†Dept. of Managerial Economics and Decision Sciences, Graduate School of Management

This paper was prepared for presentation at the 8th International Mathematical Programming Symposium, August 27-31, 1973, Stanford University, California

"Design of an Optimization System for University Use"

C. Cohen, M. Reagan, J. Stein
Northwestern University, Evanston, Illinois 60201

ABSTRACT

An optimization system for university use is quite different from MPSX or OPTIMA. This paper deals with the design and development of a Multi Purpose Optimization System (MPOS) now in use on Northwestern University's CDC 6400, which has evolved in the following stages:

a) the "stand-alone" program library - which includes over twenty linear, integer and non-linear programming algorithms,

b) the "batch" MP system - developed to permit the user to state LP/QP/IP problems in English and standard MP notation. The system, coded in FORTRAN, consists of a language pre-processor and individual overlays corresponding to the various solution algorithms,

c) the "online" MP system - designed to be used interactively at a teletype.

Our design objectives have been to stress ease of use, expandability and interactive optimization. The system's modular design allows for incorporation of new or improved algorithms. They, in turn, can be tested with an archive of test problems. Usage statistics are collected to better plan for future software development or improvement.

The advantages of interactive optimization are indicated: maximize the utility in real time between the user and the computational process, facilitate the learning of basic principles (e.g., pivoting, cuts), provide options to interrupt computation and permit the user to interpose decisions based on his experience and intuition and to spend more time on the modelling aspects of optimization.

1. Introduction

The essential role of a university computer center is to provide the university community the maximum opportunity for making use of the computer as a problem-solving or information-processing tool. A person with an optimization problem is especially eager to have dependable computer codes requiring a minimum of concern for system details outside the scope of his primary interest.

Most mathematical programming (MP) systems, such as IBM's MPSX, CDC's OPTIMA/OPHELIE/APEX, or UNIVAC's UMPIRE, are directed at the solution of very large problems stemming from corporate or industry models. An optimization system for university use is quite different from MPSX or OPTIMA. First, the software supplied by computer manufacturers is proprietary and very expensive. Second, the majority of university users have small or medium scale problems. Third, the typical user is not inclined to solve optimization problems on the computer if he has to spend a great deal of time learning about the architecture of MP systems. More often than not, their documentation is not designed for a novice user.

Our design objectives have been to stress ease of use, expandability, and interactive optimization. Our library of MP algorithms has evolved from a batch version to an online version in a little more than two years. Our system was developed to permit the user to state his problem in English and standard MP notation, to access any available linear programming (LP), integer programming (IP), or quadratic programming (QP) codes, and to edit/store/retrieve his problem files.

2. Motivation

Our motivation arose from practical needs:

- We wanted to provide a simple to use and economical LP/IP/QP system to some 500 management, engineering and economics students

enrolled in introductory optimization courses. The fact that about 150 management students are enrolled in the evening program, prompted the development of an online version which would enable them to solve optimization problems "at their office desks" if they have a terminal available.

- The choice between CDC's OPTIMA and standard "canned" programs was not very appealing. On one hand, CDC's OPTIMA is expensive to use in the classroom because of its high set-up cost. On the other hand, the standard canned programs differ in the way they accept input data and therefore force the student to follow different instructions for each algorithm. Hence, the need for a language pre-processor to create the various tableaus required by the different algorithms and act as a common input language.

- Finally, we realized that there are many pedagogical advantages in man-machine interaction. The learning process of the computational aspects of optimization is accelerated. Also, an interactive optimization system is economical in real-time. The student needs a few runs to get an answer to his problem and can, therefore, spend more time on the modelling aspects of optimization.

3. An overview of optimization software at Northwestern University

The field of mathematical software is an emerging scientific discipline and interactive applied mathematical systems have been developed in parallel with new on-line operating systems, see e.g. [25], [33]. The most notable efforts in MP systems design and implementation of the batch processing type have been those of Beale [4], Dantzig [9], [5], and Orchard-Hays [40]. For a recent survey report on the computational aspects of MP see White [41]. In this section, we briefly describe our optimization software.

3.1 The "stand-alone" library

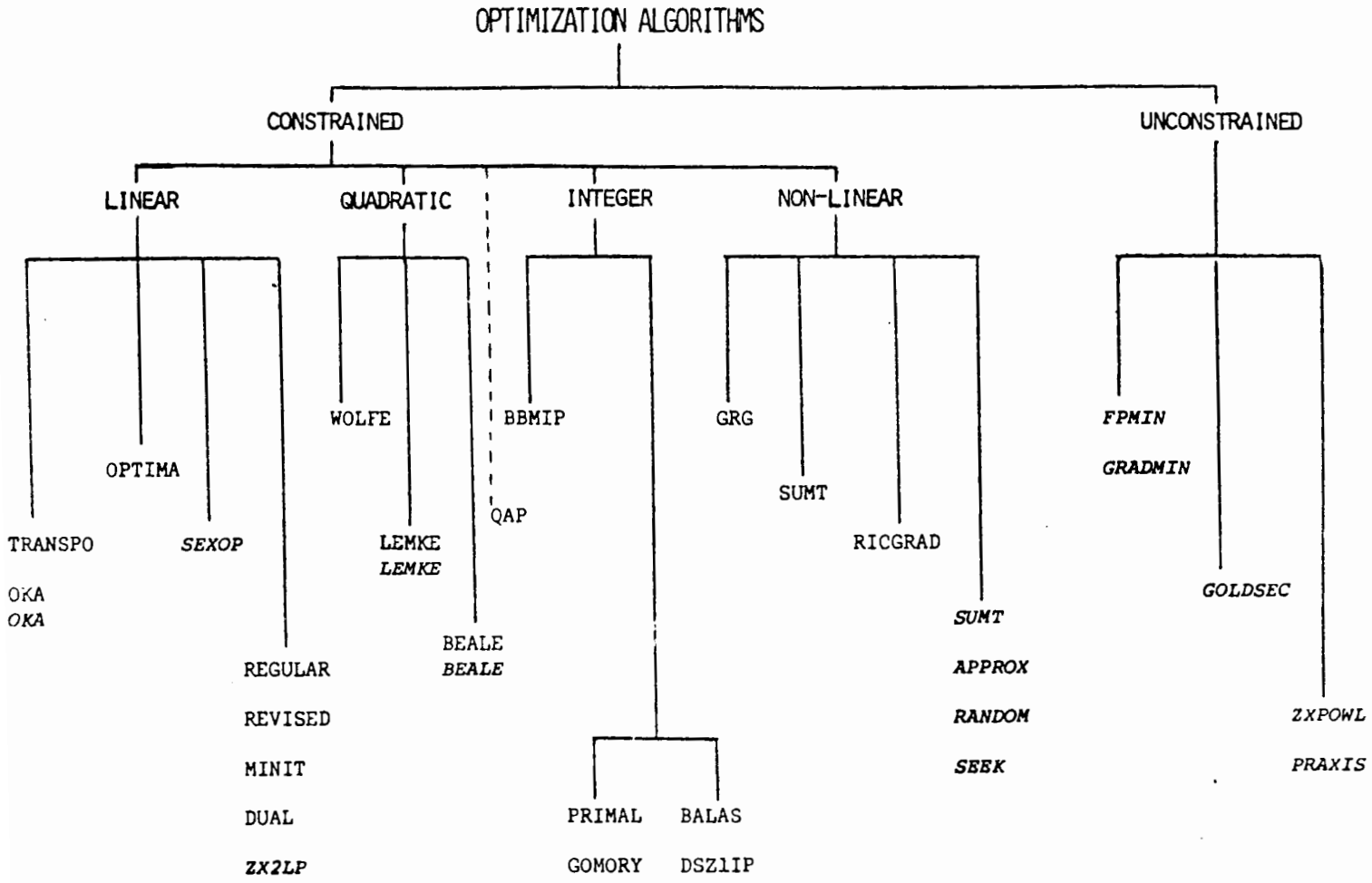
This library of "canned" programs comprises many algorithms of constrained and unconstrained optimization and is best described in Tables 1 and 2. These programs have been collected from several sources: some were developed locally, some were contributed by interested users, some were obtained from research laboratories or other university computer centers. We check them out and document them individually. The stand-alone library is tested using an archive of standard test problems (Colville [7] , Haldi [20] , Himmelblau [22] , Trauth and Woolsey [37]) or problems contributed by our own users.

3.2 The MPOS system (Multi Purpose Optimization System)

This system was developed to permit the user to state his problem in English and standard MP notation. The user has access to the LP/IP/QP algorithms. For example, he may input the following:

```
WOLFE                                    algorithm name
MAXIMIZE
X1 + X2 -0.5X1*X1 + X1*X2 - X2*X2        objective function definition
CONSTRAINTS
X1 + X2 .LE. 2                           constraints definition
2X1 + 3X2  .GE. 6
PRINT 1                                  print each iteration
OPTIMIZE
STOP                                     stop computation
```

The system consists of a language pre-processor and individual overlays corresponding to the various solution algorithms. With the exception of a few assembly language subroutines, the MP system is coded in FORTRAN. The system's modular design allows for incorporation of new or improved algorithms.



(Subroutines in italics)

TABLE I

OPTIMIZATION ALGORITHMS

Constrained optimization

Linear:

TRANSPO	Transportation algorithm (primal-dual)	[8]
OKA/*OKA*	Out-of-kilter algorithm	[8]
OPTIMA	CDC's LP system	[30]
SEXOP	Subroutines for Experimental Optimization (decomposition and GUB constraints)	[29]
REGULAR	2-phase simplex algorithm	[8]
REVISED	Revised simplex algorithm	[8]
MINIT	Primal-dual algorithm	[34]
DUAL	Dual simplex algorithm	[8]
ZX2LP	Revised simplex algorithm	[24]

Quadratic:

WOLFE	Wolfe's quadratic simplex algorithm	[26]
LEMKE/*LEMKE*	Lemke's complementary pivot algorithm	[26]
BEALE/*Beale*	Beale's quadratic programming algorithm	[26]

Integer:

BBMIP	Branch and bound mixed integer prog.	[36]
PRIMAL	Glover/Young/Harris primal cutting plane	[15,21,39]
GOMORY	Gomory's dual cutting plane algorithm	[16]
BALAS	Balas/Glover additive algorithm	[2,14]
DSZ1IP	Lemke and Spielberg direct search alg.	[27]
QAP	Graves and Whinston quadratic assignment	[17]

Non-linear:

SUMT	Fiacco and McCormick SUMT algorithm	[10]
GRG	Abadie and Carpentier generalized reduced gradient algorithm	[1]
RICGRAD	Greenstadt's ricochet gradient algorithm	[19]
SEEK	Hooke and Jeeves direct search alg.	[23]
APPROX	Griffith and Stewart linear approximation algorithm	[18]
RANDOM	Random search subroutine	[38]

Unconstrained optimization

FPMIN	Fletcher and Powell descent method	[11]
GRADMIN	Fletcher and Reeves conjugate gradient	[12]
GOLDSEC	Golden section search	[38]
ZXPOWL	Powell's method without derivatives	[31]
PRAXIS	Brent's method without derivatives	[6]

TABLE 2

3.3 The interactive MPOS system

This system is an extension of the batch system and is designed to be used interactively at a terminal. It has editing and file manipulation capabilities, such as, generating files to be processed in batch mode when output is voluminous. The interactive system can be used in two computation modes. The "direct" mode simulates the batch version. The "assisted" mode interacts with the user (see § 5).

Detailed statistics on the use of these programs permit us to better plan for future software development or improvement.

4. MPOS from the designer's view-point

Northwestern University's Vogelback Computing Center has the following hardware/software configuration:

- CDC 6400, 65K (60 bit) words central memory, 250K Extended Core Storage, SCOPE 3.3 Operating System and NU's own ONLINE time-sharing system.

The need for modularity resulted in adopting an overlay program structure for MPOS:

- The main (0,0) overlay selects the algorithm and determines if computation mode is batch or online.
- The (1,0) primary overlay is the language pre-processor. It decodes statements such as

```
5.3X1 - 2X1*X1 + X2*X1     or
X1 - 2X2  .LE. 4           or
BOUNDS
X1    .LE.    10
X2    .LE.     5
```

and performs syntax checking. Its main function is to produce a data file of a_{ij}, b_i, c_j, etc. for input to the different algorithms or to CDC's OPTIMA.

- The other primary overlays are the algorithm overlays.

Each algorithm overlay computes the exact amount of central memory required by the problem size. This dynamic storage allocation permits the loading of the system in less than 13000 decimal words. It also allows for stacking problems until the STOP command is encountered.

In all, MPOS includes some 60 subroutines. All but two are in FORTRAN. A full description of the system can be found in the system documentation manual.

5. MPOS from the user's view-point

We have tried to keep the number of control cards to a minimum, and make the input look like standard MP notation. All the user has to do is become familiar with a number of keyword commands (see Table 3).

```
***** PROBLEM NUMBER 22 *****
```

Example 2

```
    MAXIMIZE
    X1+X10
    CONSTRAINTS
    X1+X2-3 = 5
-- FATAL ERROR NUMBER   4
   MISPLACED SIGN OR NUMERIC CONSTANT

    X1+3.5X2.LE. X5+1
-- FATAL ERROR NUMBER   3
   VARIABLE ON RIGHT HAND SIDE OF RELATIONAL

    *DECK TEST21

-- PROBLEM ABORTED DUE TO FATAL ERRORS.
```

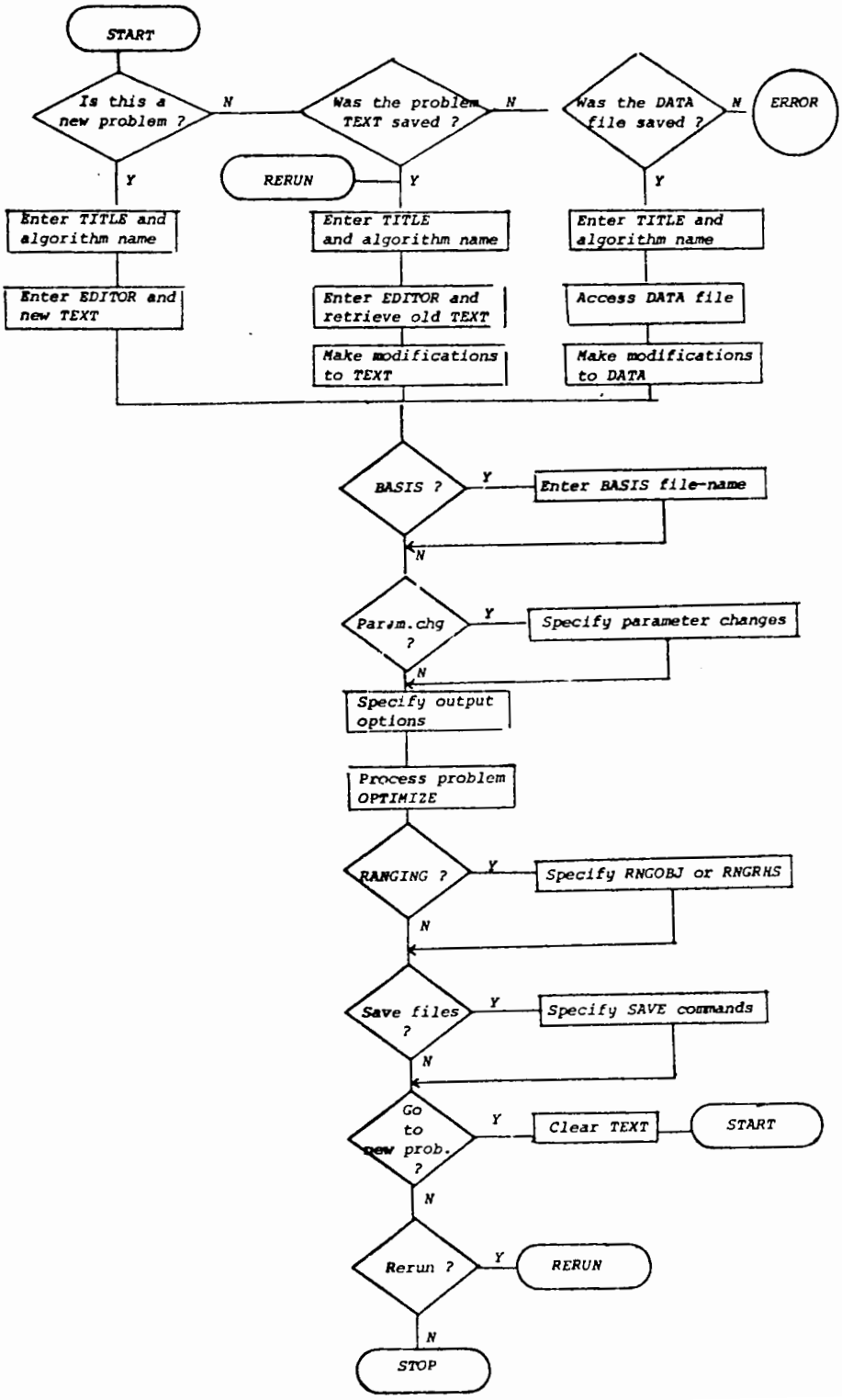
Example 3

```
EDITOR READY
? TEXT.   <-------------------------------- Enter text of QP problem
? MAXIMIZE
? X1 + X2 -0.5X1X1 - X1X2 +X1X2
? CONSTRAINTS
? X1+X2 <3
? 2X1+3X2 > 6
? OPTIMIZE
? $E   <----------------------------------- Terminate text
LAST LINE 60
? LIST.   <-------------------------------- Produce listing of text
10 MAXIMIZE
20 X1 + X2 -0.5X1X1 - X1X2 +X1X2
30 CONSTRAINTS
40 X1+X2 <3
50 2X1+3X2 > 6
60 OPTIMIZE
? REPLACE/+X1/-X2/20   <------------------- Correct '+X1' to '-X2' in line 20
20 X1 + X2 -0.5X1X1 - X1X2 -X2X2
? RL,GOQP   <------------------------------ Execute problem

   --- QP ---
   VERSION 1.5
   WOLFE'S QUADRATIC SIMPLEX



   PROBLEM NO.   1



        INITIAL TABLEAU

     BASIC    CB      VALUES
     VAR.
         9   -1.     3.0000
        10   -1.     6.0000
        11   -1.     1.0000
        12   -1.     1.0000

     Z(J)-C(J)=    -11.000


     ----- FINAL SOLUTION AT ITERATION     4


     BASIC    CB      VALUES
     VAR.
         3    0.     .40000
         8    0.     .80000          <-- Summary results
         1    0.     1.8000
         2    0.     .80000

     Z(J)-C(J)=    0.

       -1.100000 = VALUE OF OBJECTIVE FUNCTION

            .0230 CP SECONDS FOR   4 ITERATIONS

SUCCESSFUL EXECUTION
? NEW*QPPROB   <--------------------------- Save problem text on file QPPROB
? END
```

TITLE	Any alphanumeric title
REVISED...WOLFE...BBMIP	Name of the optimization algorithm
INTEGER VARIABLES	List of integer variables List of variable names
MAXIMIZE *MINIMIZE*	Objective function definition
CONSTRAINTS *CONSTRAINTS n*	Constraints definition
BOUNDS *BNDALL value* *BNDOBJ value*	Variable upper bounds Objective function upper-bound
EPSILON value *TOLERANCE value* *LIMIT n*	Perturbation factor Accuracy check Maximum number of iterations
PRINT *PRINT i*	Print initial and final tableaus Print every i-th tableau
RNGOBJ *RNGRHS*	Range on objective function coefficients Range on right-hand side coefficients
SAVEFILE file-name *GETFILE file-name* *READ file-name*	Store data file Retrieve data file Read alternate input file
PACKED *TABLEAU* *MATRIX*	Packed input Tableau input Matrix input
FORMAT	Variable format
MODIFY	Data file modifications
BASIS file-name *SAVEBASIS file-name*	Input basis Save basis
OPTIMIZE *END*	Initiate problem solution
STOP	Stop run
*	(any comment card)

TABLE 3

When solving problems interactively, the user has the option between the DIRECT and ASSISTED solution modes. In the former case, he is in fact simulating the batch version. In the latter case, he interacts with the system via questions and answers. Only summary output is printed locally. A number of examples follow the online flowchart for the ASSISTED mode.



```
          USING REGULAR

***** PROBLEM NUMBER  9 *****
```

Example 1

```
    MAXIMIZE
    ************************************************************
    *
    *  MAXIMIZE PROFIT FROM MANUFACTURE OF
    *     TABLES, CHAIRS, DESKS, AND BOOKCASES
    *
    *  VARIABLE DEFINITIONS,
    *     X1 = TABLES
    *     X2 = CHAIRS
    *     X3 = DESKS
    *     X4 = BOOKCASES
    *
    ************************************************************
      12X1  + 5X2  + 15 X3  + 10X4
    CONSTRAINTS
    **
    *  CONSTRAINTS DUE TO SALES SCHEDULE
    **
    X3 .GE. 15
    X4 .GE. 10
    **
    *  CONSTRAINTS DUE TO AVAILABLE RESOURCES
    **
    *  BOARD FEET OF TYPE I LUMBER
    5X1+X2+9X3+12X4 .LE. 1500
    *  BOARD FEET OF TYPE II LUMBER
    3X1+2X2+5X3+10X4 .LE. 1000
    *  MAN HOURS TO UTILIZE
    2X1+3X2+4X3+5X4 = 800
    OPTIMIZE


 ********************
 * PROBLEM NUMBER  9 *
 ********************

ITERATION     1   IN VAR-   4     OUT VAR-   8   -WMIN=        -765.000
ITERATION     2   IN VAR-   3     OUT VAR-   7   -WMIN=        -690.000
ITERATION     3   IN VAR-   6     OUT VAR-  10   -WMIN=        -277.500
ITERATION     4   IN VAR-   2     OUT VAR-  11   -WMIN=        0.

ENTERING PHASE II

ITERATION     5   IN VAR-   1     OUT VAR-   9   -ZMIN=         3373.00


 ********************
 * PROBLEM NUMBER  9 *
 ********************


                    SUMMARY OF RESULTS

  VARIABLE  VARIABLE     BASIC        ACTIVITY      OPPORTUNITY
    NO.       NAME     NON-BASIC       LEVEL          COST
     1        X1           B        219.0000000        --
     2        X2           B         84.0000000        --
     3        X3           B         15.0000000        --
     4        X4           B         10.0000000        --
     5      --SLACK       NB              --          3.8000000
     6      --SLACK       NB              --         33.0000000
     7      --ARTIF       NB              --         -3.8000000
     8      --ARTIF       NB              --        -33.0000000
     9      --SLACK        B        66.0000000        --
    10      --SLACK       NB              --          5.2000000
    11      --ARTIF       NB              --         -1.8000000

   MAXIMUM VALUE OF THE OBJECTIVE FUNCTION =        3373.000000

   CALCULATION TIME WAS      .0300 SECONDS FOR    5 ITERATIONS.
```

```
 - - PROBLEM NUMBER   1 - -
```
Begin new problem

```
 ENTERING EDITOR
EDITOR READY
CLEARED
? TEXT.
? *  MIN-COST NETWORK FLOW STATED AS A L.P. PROBLEM
? VARIABLE LIST
? 0A 0B BA BC AC AK CK K0
? *  SOURCE NODE = 0, SINK NODE = K, OTHER NODES=A,B,C
? MINIMIZE
? 40A + 0B + 2BA + 3BC + 6AC + AK +2CK
? CONSTRAINTS
? *  CONSERVATION OF FLOWS
? K0 -0A -0B =0
? AK +CK -K0 =0
? 0A + BA -AC -AK =0
? 0B -BA -BC =0
? BC + AC - CK =0
? K0=11
? *   CAPACITATED LINKS
? BOUNDS
? 0A<10
? 0B<8
? BA<5
? BC<10
? AC<2
? AK<7
? CK<4
? OPTIMIZE
? ZE
LAST LINE 240
? DROP
     *  MIN-COST NETWORK FLOW STATED AS A L.P. PROBLEM

                USING REGULAR
```
Enter TEXT of problem in EDITOR mode

Variable names

Objective function definition

Constraints

Bounds

Terminate TEXT

Begin execution of problem

Regular simplex is used by default

```
***** PROBLEM NUMBER  1 *****

                SUMMARY OF RESULTS

 VARIABLE  VARIABLE    BASIC        ACTIVITY      OPPORTUNITY
   NO.       NAME    NON-BASIC       LEVEL          COST
    1        0A          B          3.0000000        --
    2        0B          B          8.0000000        --
    3        BA          B          4.0000000        --
    4        BC          B          4.0000000        --
    5        AC         NB             --          5.0000000
    6        AK          B          7.0000000        --
    7        CK          B          4.0000000        --
    8        K0          B         11.0000000        --
    9      --ARTIF      NB             --          7.0000000
   10      --ARTIF       B          0.0000000        --
   11      --ARTIF      NB             --          3.0000000
   12      --ARTIF      NB             --          5.0000000
   13      --ARTIF      NB             --          2.0000000
   14      --ARTIF      NB             --         -7.0000000
   15      --SLACK       B          7.0000000        --
   16      --SLACK      NB             --          1.0000000
   17      --SLACK       B          1.0000000        --
   18      --SLACK       B          6.0000000        --
   19      --SLACK       B          2.0000000        --
   20      --SLACK      NB             --          2.0000000
   21      --SLACK       B          0.0000000        --

   MINIMUM VALUE OF THE OBJECTIVE FUNCTION =          55.000000

   CALCULATION TIME WAS      .0220 SECONDS FOR    7 ITERATIONS.

 PROBLEM COMPLETE, TYPE
      STOP  TO TERMINATE RUN,
   OR  EDIT  TO MODIFY PROBLEM.
? EDIT
```
Problem is to be EDITed

```
 - - PROBLEM NUMBER   2 - -
```
Now solve the same problem as an I.P.

```
 ENTERING EDITOR
EDITOR READY
CLEARED
 24 LINES READ
? 5.BBMIP
? 6.INTEGER LIST
? 7.0A 0B BA BC AC AK CK
?
DROP
                USING BBMIP
```
Insert BBMIP command

Insert INTEGER list

Insert list of INTEGER var.

Execute problem

Branch and Bound Mixed Integer Programming

```
***** PROBLEM NUMBER  2 *****


        NO. OF ROWS INCLUDING OBJ. FN.      7
        NO. OF COLS. INCLUDING RIGHT-HAND-SIDE      9
        NO. OF INT. VARIABLES      7

      ITERATION      1: OUT ROW-    6, IN COL-    8, -Z=-0.
      ITERATION      2: OUT ROW-    1, IN COL-    2, -Z=  11.000
      ITERATION      3: OUT ROW-    2, IN COL-    6, -Z=  22.000
      ITERATION      4: OUT ROW-    3, IN COL-    2, -Z=  33.000
      ITERATION      5: OUT ROW-    4, IN COL-    3, -Z=  44.000
      ITERATION      6: OUT ROW-    1, IN COL-    1, -Z=  47.000
      ITERATION      7: OUT ROW-    2, IN COL-    3, -Z=  55.000


PROBLEM NUMBER  2


--- CONTINUOUS SOLUTION ---

OBJECTIVE FUNCTION=    55.00000      AT ITERATION       7. TIME=     .052

                SUMMARY OF RESULTS

  VARIABLE      BASIC/   INTEG/          ACTIVITY          OPPORTUNITY
  NO.   NAME    NON-B    CONTIN            LEVEL              COST
    1   0A        B        I             3.0000000          --
    4   BC        B        I             4.0000000          --
    7   CK        B        I             4.0000000          --
    3   BA        B        I             4.0000000          --
    0   --ARTIF   B        C            -0.0000000          --
    8   K0        B        C            11.0000000          --
    2   0B       NB        I             8.0000000         -1.0000000
    5   AC       NB        I               --               5.0000000
    6   AK       NB        I             7.0000000         -2.0000000

TOLERANCE SET AT    60.50000      AT ITERATION      7

CONTINUOUS SOLUTION IS INTEGER SOLUTION.

OPTIMALITY ESTABLISHED AT ITERATION       7

----END OF PROBLEM.  TOTAL TIME WAS     .095 SECS.

PROBLEM COMPLETE, TYPE
     STOP  TO TERMINATE RUN,
  OR  EDIT  TO MODIFY PROBLEM.
? STOP
```
Terminate run

6. Evaluation of the system and conclusions

In the previous sections we described a software system designed to solve linear, integer and quadratic optimization problems using well established algorithms. Because of its simple structure and repertoire of algorithms, MPOS has become one of the most widely used applications programs at Northwestern University. As a result, students feel less "intimidated" by non-linear optimization models and they find it practical to experiment with extensions of LP models. Our emphasis on modularity facilitates expandability. The system is flexible enough to incorporate new algorithms. A student or researcher can "plug" into our system a new algorithm module without much concern for input/output and evaluate it with our archive of test problems.

Our system has proven to be economical from the view-points of usage and maintenance. The user needs few runs to get a solution to his problem. If, for example, an integer programming algorithm fails, the user can switch to another one by changing one card or editing a line. Because the interactive system needs a very small amount of central memory (5000 words) to load, response times are very good. Maintenance of the system is greatly simplified by the overlay structure which lends itself to "decentralized programming" activities. Good internal documentation and machine independence (of the batch version) facilitate transportability.

It is quite evident that our problem sizes are limited by the capacities of central memory and Extended Core Storage. Our system is not designed to compete with the new large-scale MP systems, such as those under development at Stanford's System Optimization Laboratory or MIT's NBER/Computer Research Center in Economics and Management Science. Users with large problems turn to APEX I and II, a commercial LP/MIP system we just obtained from CDC.

The reliability of the algorithms depends on the efficiency of the numerical methods used to implement them. For example, to speed computation time we perform row elimination procedures in assembly language. We are also studying the adoption of new matrix decomposition techniques (Bartels and Golub [3], Saunders [35]) to improve numerical accuracy.

Too often, system designers have concentrated on effective utilization of hardware and software resources with little attention paid to man/machine dialogue. Stated simply, these systems have been designed from the inside, out. To be user-effective, systems must be designed from the outside, in, because computer users come from a spectrum of individuals with different levels of knowledge and different kinds of problems. The value of this MP system will be judged by the users of MPOS and by the amount by which MPOS reduces the total time it takes a person to solve an optimization problem.

This system represents the first step toward our goal of a general interactive optimization system designed to "graft" the computer to students or researchers involved in optimization modelling. Existing algorithms are highly formalized and do not allow much human intervention and interaction. We believe that an interactive optimization system fulfills a great need in both research and teaching. Faced with a LP problem with multiple objective functions, the decision maker learns to recognize good solutions and the relative importance of (competing) objectives through sentivity analysis performed online. In fact, phases of computation alternate with phases of decision.

Although the present system excludes NLP algorithms, there are instances where the user may change solution strategy through the use of different algorithms. For example, starting with a cutting-plane method and terminating with a branch and bound. There are a number of practical and theoretical problems in non-linear optimization where mixed-strategy can lead to new insights: convergence properties of algorithms, interaction of algorithms and criteria for switching from one to

the other, solution of non-convex or ill-structured problems, best matching of an algorithm to a given class of problems. As more experience is gained with our existing system, we will have a better perspective for designing the interactive non-linear optimization system.

In closing, we can state that this operational system has pedagogical advantages, facilitates man-machine problem-solving in that it brings the union of optimization and computing. It is responsive to the user needs and background and can be used as an inexpensive tool for training users in the power and pitfalls of various optimization techniques.

Acknowledgements

We wish to thank Joseph Yozallinas for his programming efforts in the initial stages, and Robert Grierson and Irma Waye for their recent programming assistance.

References

1. Abadie, J. and J. Carpentier, "Generalization of the Wolfe reduced gradient method to the case of nonlinear constraints," in [13].

2. Balas, E., "An additive algorithm for solving linear programs with 0-1 variables," Operations Research, 13, (1965), pp. 517-576.

3. Bartels,R.H., and G.H. Golub, "The simplex method of linear programming using LU decomposition," Comm. of the ACM, 12, May 1969.

4. Beale, E.M.L. and J.A. Tomlin, "Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables," Proc. 5th Intl. Conference on Operations Research, Venice (1969).

5. Bonzon, P.E., "MPL: an appraisal based on practical experiments," Computer Science Report No. 72-267, Stanford University, California (1972).

6. Brent, R.P., "Algorithms for finding zeros and extrema of functions without calculating derivatives," Computer Science Report CS71-198, Stanford University, California (1971).

7. Colville, A.R., "A comparative study of nonlinear programming codes,"IBM N.Y. Scientific Center Report 320-2949, June 1968.

8. Dantzig, G., Linear Programming and Extensions, Princeton University Press (1963)

9. Dantzig, G., et al.,"MPL - A mathematical programming language," Computer Science Report No. 119, Stanford University, California (1968).

10. Fiacco, A.V. and G. McCormick, Nonlinear Programming Sequential Unconstrained Minimization Techniques, J. Wiley (1968).

11. Fletcher, R. and M.J.D. Powell, "A rapidly convergent descent method for minimization," Computer Journal, 6, pp. 163-168,(1963).

12. Fletcher, R. and C.M. Reeves, "Function minimization by conjugate gradients," Computer Journal, 7, pp. 149-154, (1964).

13. Fletcher, R., Ed., Optimization, Academic Press, (1969).

14. Glover, F. and S. Zionts, "A note on the additive algorithm of Balas," Operations Research, 13, (1965), pp. 546-549.

15. Glover, F., "A new foundation for a simplified primal integer programming algorithm,"Operations Research, 14, (1966), pp. 1045-1074.

16. Gomory, R.E., "All-integer integer programming algorithm," IBM Research Center Report RC-189, Jan. 1960; also in Muth and Thompson (eds.), Industrial Scheduling, Prentice-Hall (1963), pp. 193-206.

17. Graves, G. and A. Whinston, "An algorithm for the quadratic assignment problem," Management Science, 17, pp. 453-471, (1970).

18. Griffith, R.E. and R.A. Stewart, "A nonlinear programming technique for the optimization of continuous processing systems," Management Science, 7, pp. 379-392, (1961).

19. Greenstadt, J.C., "A ricocheting gradient method for non-linear optimization," Journal of SIAM, Appl. Math, 14, pp. 429-445, (1966), also IBM Program Library No. 360D-15.3.001.

20. Haldi, J. "25 integer programming test problems," Working Paper No. 43, Graduate School of Business, Stanford University, California (1964).

21. Harris, P.M.V.,"An algorithm for solving mixed integer linear programs," Operations Research Quarterly, 15, (1964), pp. 117-132.

22. Himmelblau, D.M., Applied Nonlinear Programming, McGraw Hill (1972).

23. Hooke, R., and T.A. Jeeves, "Direct search solution of numerical and statistical problems," Journal of the ACM, 8, (1961).

24. IMSL, International Mathematical and Statistical Libraries, CDC 6000 Library 3, Houston, Texas.

25. Klerer, M. and J. Reinfelds, Eds., Interactive Systems for Experimental Applied Mathematics, Academic Press (1968).

26. Kunzi, H.P. and W. Krelle, Nonlinear Programming, Blaisdell,(1966).

27. Lemke, C. and K. Spielberg,"Direct search 0-1 and mixed integer programming," Operations Research, 15, (1967), also Program Library No. 360 D-15.2.001, (1968).

28. Lemke, C.E. and J.T. Howson, Jr., "Equilibrium points of bimatrix games," Journal of SIAM, 12, pp. 413-423, (1964).

29. Marsten, R.M., Users Instructions for SEXOP, Subroutines for Experimental Optimization, Dept. of Industrial Engineering and Management Science, Northwestern University, (1972).

30. OPTIMA, CDC Reference Manual, Publication No. 60207000, 1968.

31. Powell, M.J.D., "An efficient method for finding the minimum of a function of several variables without calculating derivatives," Computer Journal, 7, pp. 155-162, (1964).

32. Ravindran, A., Algorithm 431: "A computer routine for quadratic and linear programming problems," CACM, 15, pp. 818-820, (1972).

33. Rice, J., Ed., Mathematical Software, Academic Press, (1971).

34. Salazar, R.C. and S.K. Sen, "MINIT: minimum-iteration algorithm for linear programming," CACM Algorithm 333 (1968).

35. Saunders, M.A.,"Large scale linear programming using the Cholesky factorization," Computer Science Report No. 72-252, Stanford University, California (1972).