

LEVEL II



AD A095020

NPS52-80-006

NAVAL POSTGRADUATE SCHOOL

Monterey, California



DTIC
ELECTRONIC
FEB 13 1981

ATHENA:
USERS MANUAL FOR INTERACTIVE ANALYSIS
OF LARGE-SCALE OPTIMIZATION MODELS
by
Gordon H. Bradley
Gerald G. Brown
Panagiotis I. Galatas
April 1980

Approved for public release; distribution unlimited.

Prepared for:
Naval Postgraduate School
Monterey, California 93940

BBC FILE COPY

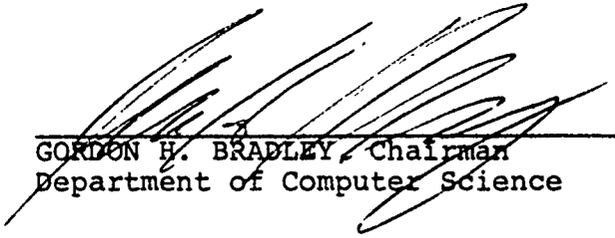
81 2 13 04

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rear Admiral J. J. Ekelund
Superintendent

Jack R. Borsting
Provost

This report prepared by:


GORDON H. BRADLEY, Chairman
Department of Computer Science


GERALD G. BROWN
Department of Operations Research


PANAGIOTIS I. GALATAS

Reviewed by:

Released by:


MICHAEL G. SOVEREIGN, Chairman
Department of Operations Research


WILLIAM M. TOLLES
Dean of Research

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NPS52-80-006	2. GOVT ACCESSION NO. AD-A095020	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) ATHENA: Users Manual for Interactive Analysis of Large-Scale Optimization Models.		5. TYPE OF REPORT & PERIOD COVERED 9 Technical Report
7. AUTHOR(s) Gordon H./Bradley, Gerald G./Brown/ Panagiotis I./Galatas		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		8. CONTRACT OR GRANT NUMBER(s)
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) 12 55		12. REPORT DATE 11 Apr 80
		13. NUMBER OF PAGES
		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Large-scale optimization, Linear programming, Linear program report writing, Mixed integer optimization, Interactive model analysis, Matrix generation in linear programming		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) Analyses of solutions for large-scale optimization models are very difficult without effective computer aids. Solution reports may require weeks to design, implement and produce with conventional report writing systems. The reports produced are voluminous, often exceeding 100,000 printed lines, and are thus quite awkward to access manually. Timely and economic analysis of solutions to large models is further hindered by inflexible and costly report writing software and procedures. ATHENA has		

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

been developed to allow extremely efficient *immediate interactive* storage and analysis of the solution file from *any* optimization system. ATHENA is easy to learn and use; "user friendly" features are provided which can preemptively assess the potential cost and implications of each request for solution information, assist the confused user, and provide the required solution information with very fast response time. The user is provided with extensive "search under mask" and "compound logical relational" constructs, as well as the capability to quickly diagnose suspicious model symptoms, and to format and issue offline reports. ATHENA is implemented in *portable* FORTRAN, with a parser and interpreter easily modified and expanded to suit particular hardware environments and user demands. The system has been initially designed and tuned for large-scale problems with up to 30,000 rows and columns. Live test demonstrations show that the system exhibits very fast response time in actual use. This report presents a users manual for the prototype ATHENA query language, an error message directory, and a description of interface and extension provisions.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

TABLE OF CONTENTS

I.	INTRODUCTION	I-1
II.	SOFTWARE DESCRIPTION	II-1
	A. GENERAL	II-1
	1. Parser - Code Generator	II-1
	2. Input	II-1
	3. Interpreter	II-1
	B. LANGUAGE	II-1
	1. Host Language	II-1
	2. Query Language	II-3
	C. DATA STRUCTURES	II-4
	1. Preview	II-4
	2. SPARSE Data Structure	II-6
	3. SUPERSPARSE Data Structure	II-11
III.	USER'S MANUAL	III-1
	A. INTRODUCTION	III-1
	B. QUERY LANGUAGE	III-1
	1. Control Queries	III-2
	a. VERIFY	III-2
	b. NOVERIFY	III-2
	c. PROMPT	III-2
	d. NOPROMPT	III-3
	e. H Any String	III-3
	f. * Any String	III-3
	g. END	III-3

A

2. Command Queries	III-3
a. TYPE Field	III-4
b. SELECT Field	III-6
c. MASK Field	III-7
d. CONDITION Field	III-8
e. PRINT OPTION Field	III-11
3. The SET Command	III-12
C. ERROR MESSAGES	III-14
D. LIMITATIONS - EXTENSIONS	III-17
E. SYSTEM INTERFACE	III-19
1. Preview	III-19
2. Input File	III-20
3. Packed File	III-21
F. EXAMPLE OF SYSTEM USE	III-22
1. Obtaining the Unpacked File	III-22
2. Using the System Under CP/CMS	III-23

LIST OF REFERENCES

INITIAL DISTRIBUTION LIST

I. INTRODUCTION

The generation, solution and analysis of large scale mathematical programming models presents significant problems in efficient data handling and interpretation. For a typical large scale model the printed output may exceed 100,000 lines. It is very awkward and time consuming for the user to examine that much paper to extract the information he needs. Space is also required for archive storage of such reports and it is very difficult to provide routine access to these old solution files.

Over the past several years, faculty and students at the Naval Postgraduate School and the University of California at Los Angeles have been cooperatively developing theory and algorithms to solve large scale linear, nonlinear and integer optimization problems. This research has been rewarded by the development of many software systems to solve large scale optimization problems. ATHENA is part of that development and has been designed to satisfy a pressing need to be able to quickly and easily analyze solutions to large scale optimization models.

The research in large scale optimization at the Naval Postgraduate School has concentrated on providing economic solutions to current Department of Defense problems. One such project that was going on concurrent with the development of ATHENA was a large, medium-range capital budgeting problem that required the solution of a mixed integer programming problem with 11,687 constraints and variables [8]. ATHENA was used successfully with this project, and performance of ATHENA on this problem is reported below.

ATHENA has been developed to handle output from large scale optimization problems by enabling the user to get the information he needs interactively through a computer terminal and by economically storing the large files in packed form in low cost media. The features of the system are summarized:

1. Quick and accurate answers to simple questions that are tedious and error prone to address manually, e.g.

How many of a set of variables are ≤ 1.0 ?

How many are greater than 0?

What variables are in a specified range?

What constraints are satisfied exactly? Etc.

2. In large scale mathematical programming models the names of rows and columns are customarily constructed systematically so that groups of variables with relationships in the real world have similar names. Using the system one can have automatic, easy and accurate answers for many interesting properties of these groups. (For instance, the average value of all the variables whose names begin with X, etc.)

3. ATHENA can also be used as a basis for a simple, fast-response report writer.

4. The system provides very compact computer storage: the solution file is typically packed into 1/10 of its original volume. For example, a solution file from the IBM MPS/360 package [6] for a linear programming model with 12,000 rows and columns occupies 1.5 magnetic tapes 2400 feet long at 800 BPI in original unblocked form. In packed form the solution occupies approximately 31 feet of magnetic tape.

5. There is systematic and economic file organization with easy and accurate access.

6. A file structure for multiple runs of the problem for comparisons is available.

7. The system requires modest resources (memory, compute time) in a time-sharing environment.

8. The system is portable and allows easy change or expansion. It is implemented with an open-ended syntax analyzer in FORTRAN.

ATHENA was inspired by a similar system developed for the Department of Energy by O'Neil and Sanders [9] called PERUSE. PERUSE was developed to aid in the analysis of large linear programming energy models. A study of the needs of Naval Postgraduate School students and faculty showed that additional capacities beyond those in PERUSE were necessary to support current and future research in large scale optimization. In addition to the standard MPS output, it was determined that ATHENA should support the experimental optimization system XS [4]; the output of this system contains in addition to standard MPS output, upper and lower penalties that implement the 'elastic formulation' of linear models that is unique to XS. ATHENA also had to support the use of a preprocessor PREP [3] that reformulates the original optimization problem to an equivalent reduced problem with fewer rows and/or columns. A study of past and current modeling efforts at the Naval Postgraduate School identified additional commands that would help in the analysis of large models.

ATHENA was designed to be as much as possible a direct extension of PERUSE. Almost all the commands and options of PERUSE have been included with the identical names and syntax whenever possible. A summary of the extensions is listed under the section LIMITATIONS - EXTENSIONS, of the user's manual.

II. SOFTWARE DESCRIPTION

A. GENERAL

The whole system consists of 3 basic subsystems (see Figure 1).

1. PARSER - Code Generator

This subsystem accepts as input a Query, parses it examining the syntax according to the productions of the Query Language and generates the corresponding internal code or gives information for syntax errors. The internal codes for each Query are shown in the program list.

2. INPUT

This subsystem accepts as input either (1) an unpacked solution file in 'standard' format which it packs and saves for future reference, or (2) an L.P. solution file in packed form from a previous session.

3. INTERPRETER

This subsystem, using the code generated from the PARSER, searches the packed solution file and prints out the information requested.

B. LANGUAGE

1. Host Language

The system has been developed in a portable subset of FORTRAN IV. FORTRAN was chosen for the following reasons:

a. FORTRAN is a general language available at almost any computer installation, so the system can be used with

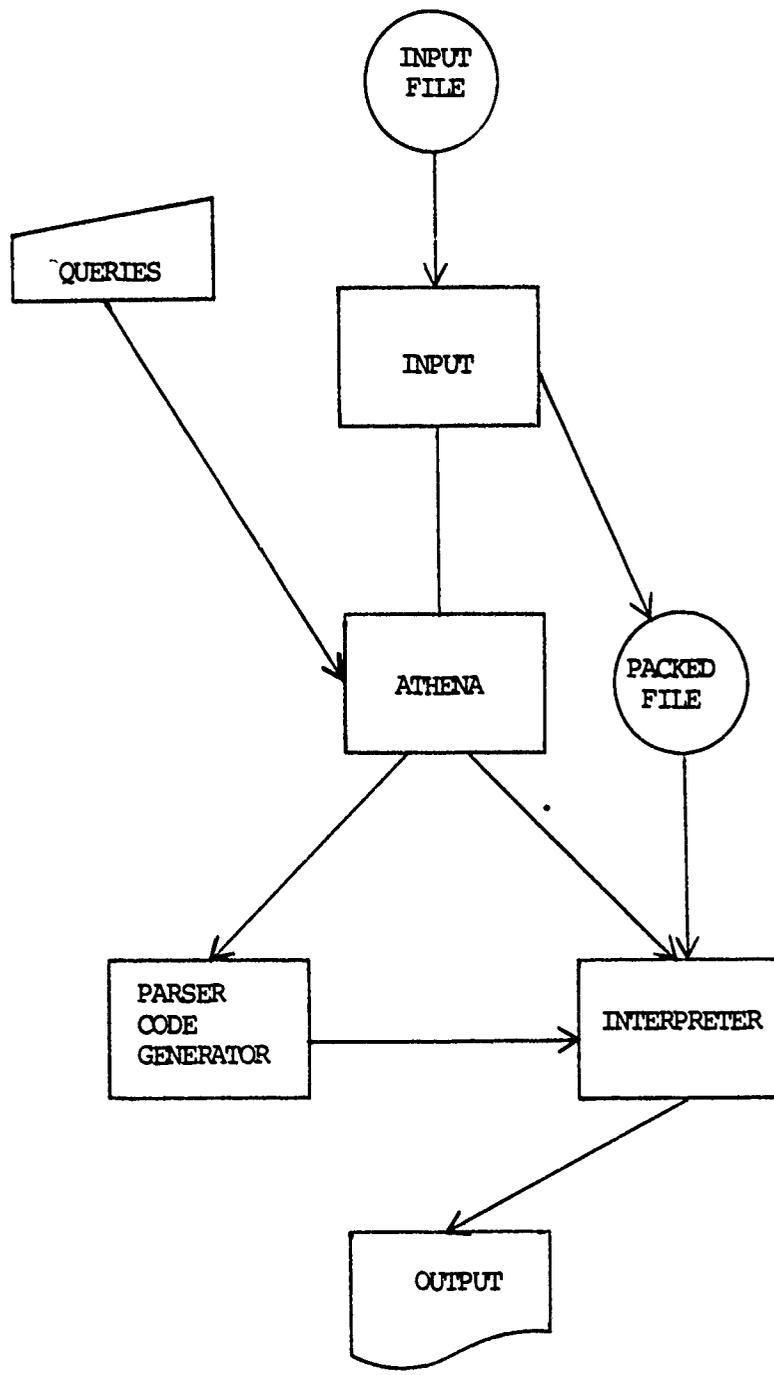


FIGURE 1. System ATHENA

any contemporary hardware. Non-IBM systems may require some program modifications.

b. Since FORTRAN is a high level language, the development time was low.

c. The response time for each Query is acceptable and there is little need for faster responses or enhanced efficiency.

d. Extensions and changes of the system can be easily implemented.

e. There is good system support for FORTRAN. In particular, ATHENA was developed with the FORTRAN H (Extended) compiler.

2. Query Language

The set of acceptable Queries is divided into two main categories:

a. Control Queries

Control Queries provide commands to the system to perform specific tasks, but usually do *not* use the solution file. Examples of Control Queries are those that accept comments for self-documentation of the output, print headings for the output, terminate the use of system, etc.

b. Command Queries

Command Queries use the solution file to extract the information asked for. Each Command Query consists of various fields separated by at least one blank. Some of the fields are optional, while others are required. An internal code number is generated by parsing the Command Query for

each field depending upon the analysis of that field and the previous fields in the Command Query.

The code generated by the PARSER is executed by the INTERPRETER, which consists of a set of programs (subroutines) activated by the code numbers, to get the required information from the file.

C. DATA STRUCTURES

1. Preview

There are some observations about the solution file of a linear program, especially of a large scale one, that lead to the use of a special data structure for storing a solution file in less memory space than it would otherwise require.

For each row or column the following information is usually included in the solution file.

NUMBER of row or column.

NAME, usually 6-8 alphanumeric characters.

STATUS, usually 2 characters, e.g., BS for BASIC, LL for LOWER LIMIT, etc.

ACTIVITY LEVEL, for each row or column.

SLACK ACTIVITY for rows or INPUT COST for columns.

LOWER LIMIT

UPPER LIMIT

DUAL ACTIVITY for rows or REDUCED COST for columns.

UPPER PENALTY and

LOWER PENALTY for the elastic linear programming system, XS [4].

There is redundant information in each record. Some of these redundancies are the following:

The explicit number for each row or column may be represented implicitly by the ordinal position of the row or column in the file. Rows usually precede columns in solution files.

A large number of the ACTIVITY LEVEL values will be zero. The same is true for the SLACK ACTIVITY, LOWER LIMIT, DUAL ACTIVITY and PENALTY values.

In many cases there will not be LOWER or UPPER LIMITS or PENALTIES.

PENALTIES in some cases may be infinite.

When the status of a row or column is 'fixed', then ACTIVITY LEVEL, LOWER and UPPER LIMITS are all the same number.

Each row or column can be in only one of its possible states.

Moreover, analysts who have experience with large scale Linear Programming have observed that most of the numbers of the solution file are the same. For example, most of the numbers for LIMITS are the same for a large number of rows or columns. For purposes of analysis, it is rarely necessary to have more than five decimal digits of precision for problem values. Indeed, some large problems cannot be solved with even this degree of significance. Accordingly, IBM single precision REAL*4 representation

is adequate for our purposes. Conversion to REAL*8 extended precision requires trivial program modifications.

Based on the above observations, two types of data structures for storing the solution file have been developed. The first one (SPARSE) exploits the redundant information in each individual record. The second (SUPERSPARSE) stores each distinct real number only once for the entire file. It is the responsibility of the user to select the data structure type that is appropriate for each solution file. SUPERSPARSE is probably superior with problems for which less than half of all non-zero coefficients possess distinct real values. What follows is a detailed description of these two data structures.

2. SPARSE Data Structure

The entire solution file is stored in contiguous memory (8-bit bytes) as a one-dimension array called SOLFIL, in the following way:

a. The first 16 bytes (four 4-byte words) are used to keep information for:

(1) The size of the file in 4-byte words.

(2) The type of data structure used to pack the file (SPARSE or SUPERSPARSE).

(3) The number of rows and columns of the file.

b. For each row or column, 12 sequential bytes are required, organized as follows (see Figure 2).

(1) The first 8 bytes hold the name of the row or column left justified, one character per byte.

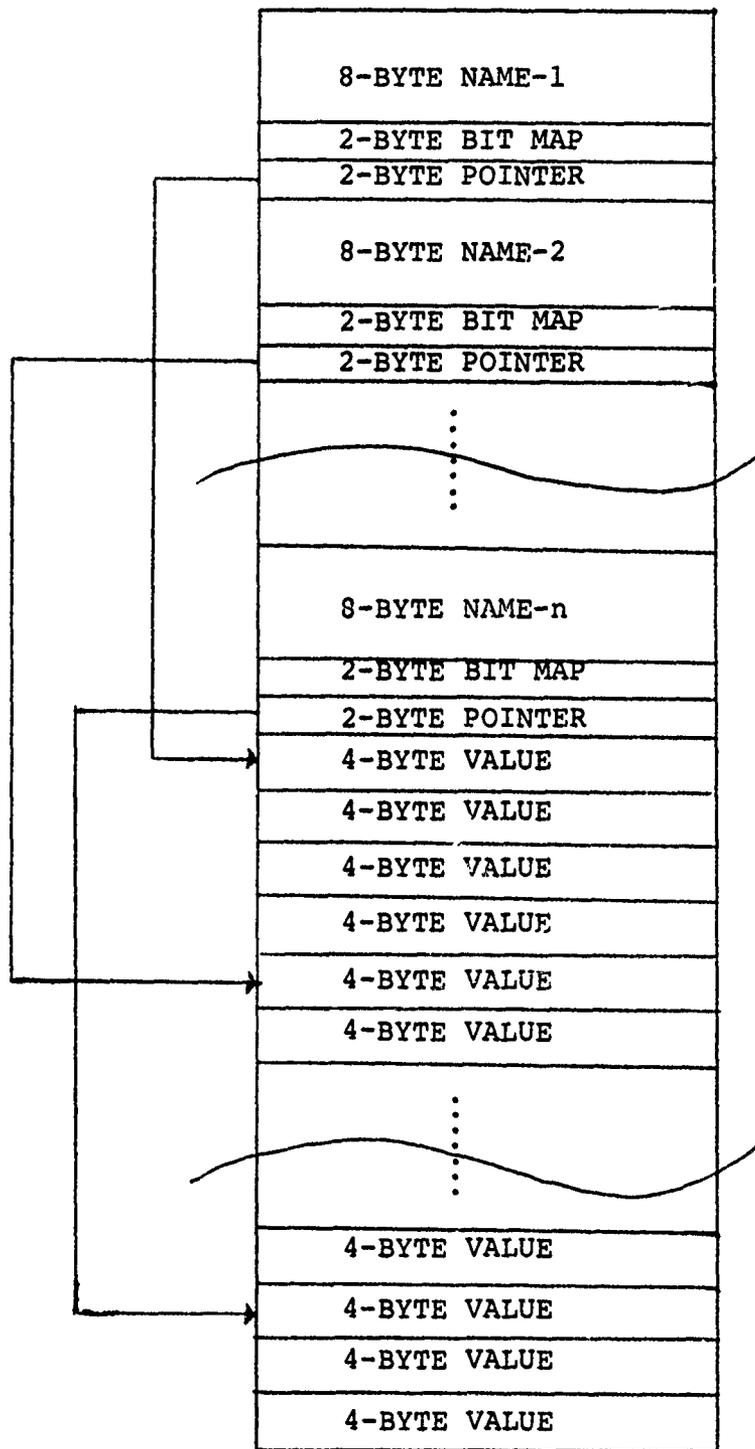


FIGURE 2. SPARSE Data Structure

(2) The next 2 bytes are used (as 16 bits) to represent various characteristics associated with that row or column.

(3) The last 2 bytes are used as a pointer to the first number stored from the current record.

c. The 16 bits from (2) above are organized in 4 groups of 4, 7, 4 and 1 bits, respectively, taken from higher to lower order.

The first group of 4 bits represents the status of the current row or column.

<u>BIT</u>	<u>PATTERN</u>	<u>STATUS</u>
0 0 0 0		IN (INFEASIBLE)
0 0 0 1		BS (BASIC)
0 0 1 0		LL (LOWER LIMIT)
0 0 1 1		UL (UPPER LIMIT)
0 1 0 0		EQ (FIXED)

The following status indicators are reserved for use with the program PREP [3]

0 1 0 1	VC (VOID COLUMN)
0 1 1 0	SC (SINGLETON COLUMN)
0 1 1 1	FC (FIX COLUMN)
1 0 0 0	BC (BOUND CHANGED)
1 0 0 1	VR (VOID ROW)
1 0 1 0	SR (SINGLETON ROW)
1 0 1 1	RR (REDUNDANT ROW)
1 1 0 0	FR (ROW FIXES VAR. AT BOUND)
1 1 0 1	ER (DOUBLETON EQUATION)

1 1 1 0

TR (TIGHTEN RANGE)

1 1 1 1

PP Reserved for PREP[3]

The next group of 7 bits represents the characteristic of zero or nonzero values for the record.

<u>BIT NO:</u>	<u>BIT VALUE:</u>	<u>CHARACTERISTIC</u>
11	0	ACTIVITY LEVEL NONZERO
	1	ACTIVITY LEVEL ZERO
10	0	SLACK/COST NONZERO
	1	SLACK/COST ZERO
9	0	LOWER LIMIT NONZERO
	1	LOWER LIMIT ZERO
8	0	UPPER LIMIT NONZERO
	1	UPPER LIMIT ZERO
7	0	DUAL/RED. COST NONZERO
	1	DUAL/RED. COST ZERO
6	0	LOWER LIMIT EXISTS
	1	LOWER LIMIT DOESN'T EXIST
5	0	UPPER LIMIT EXISTS
	1	UPPER LIMIT DOESN'T EXIST

The next group of 4 bits represents the characteristics for PENALTIES.

<u>BIT PATTERN</u>	<u>UPPER PENALTY</u>	<u>LOWER PENALTY</u>
0 0 0 0	ZERO	ZERO
0 0 0 1	ZERO	NUMBER
0 0 1 0	ZERO	INFINITY
0 0 1 1	NUMBER	ZERO

0 1 0 0	NUMBER	NUMBER
0 1 0 1	NUMBER	INFINITY
0 1 1 0	INFINITY	ZERO
0 1 1 1	INFINITY	NUMBER
1 0 0 0	INFINITY	INFINITY

The rest of the bit permutations are not used.

The last (0 bit) is used by the interpreter to mark the active and nonactive records when the user uses the ACTIVE or DEACTIVE commands to avoid searching of the entire file.

All the above groups of bits are stored together as a 16 bit binary number, which is stored in 2-byte halfword. ATHENA has provisions for the use of 16 bit halfwords representing absolute magnitudes of 0 - 65535, and can extract any component bits of the halfwords as necessary. (In this sense, the usual *signed* magnitude of IBM/360 halfword integers is ignored.)

d. The (nonzero, noninfinite) number values which must be stored are located immediately after all the information above. If the file represents a problem with M rows and N columns, then location INDEX - where $INDEX = (N+M) * 3 + 4 + 1$ - of the SOLFIL array is the first eligible location for storing number values. The value of INDEX is kept in a 2-byte pointer associated with each row and indicates for that row the location of the first value stored. The sequence for storing these numbers for each row is:

ACTIVITY LEVEL, SLACK/INPUT COST, LOWER LIMIT, UPPER LIMIT,
DUAL/REDUCED COST, UPPER PENALTY, LOWER PENALTY.

3. SUPERSPARSE Data Structure

This type of data structure takes advantage of the fact that in most problems many number values in the solution file are the same. Each distinct value is stored only once and a 2-byte pointer is used to access this value when needed. This is the only difference from the SPARSE representation (see Figure 3).

The array with the packed solution file is now separated into 3 parts:

- a. The first part is exactly the same as in SPARSE.
- b. The second part is substantially the same with the following differences:

- (1) It consists of 2-byte halfwords instead of 4-byte words.

- (2) Each halfword is a pointer to the third part of the array where the distinct number values are stored.

- c. The third part consists of a pool of 4-byte words, each representing a distinct real number value. The pointers to the distinct real number values are relative addresses in the real number pool, so a file which is packed with a different array size can be used with the current pointers providing the array size is large enough to hold the file.

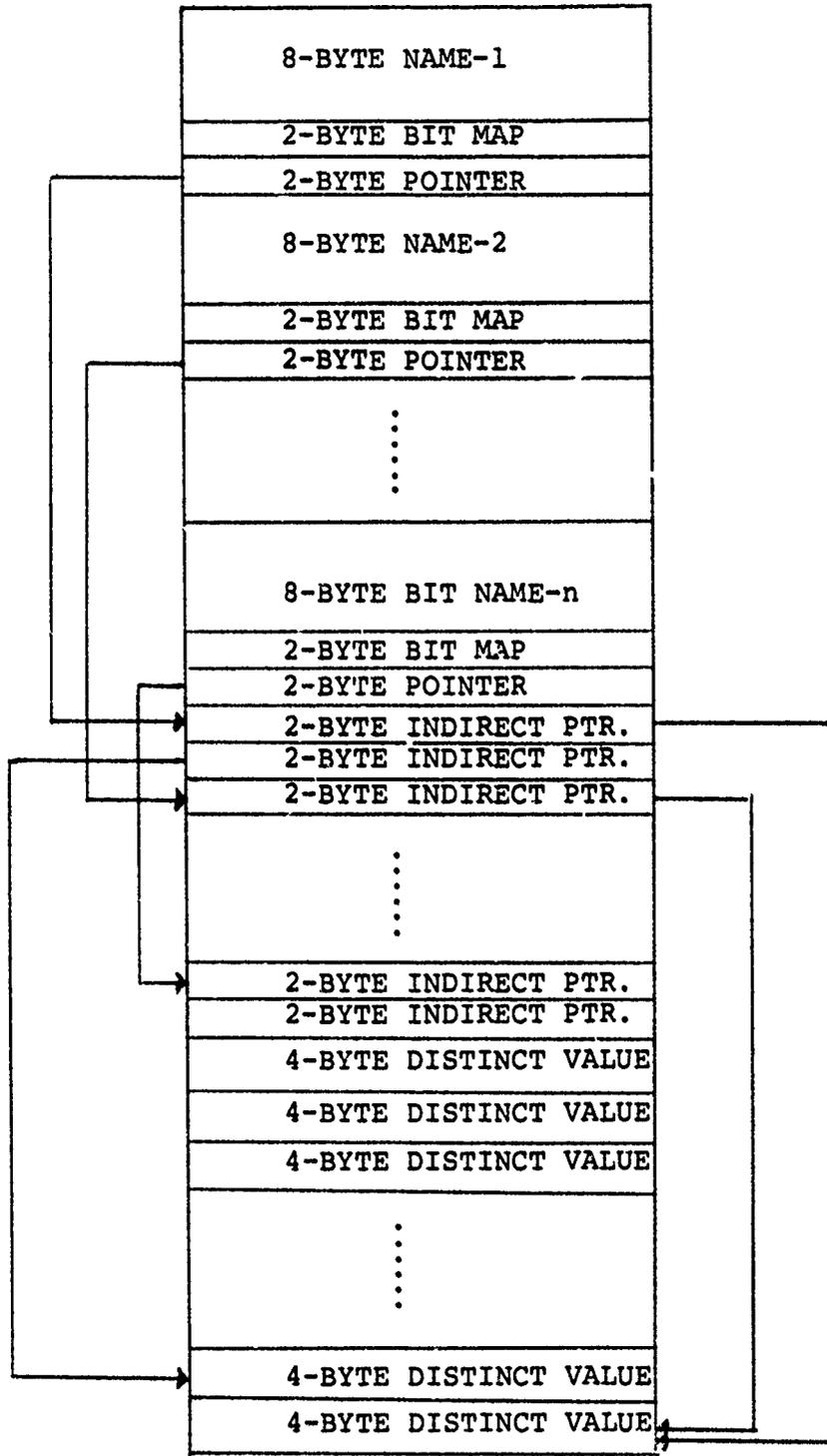


FIGURE 3. SUPERSPARSE Data Structure

III. USER'S MANUAL

A. INTRODUCTION

The system ATHENA is a set of programs which accepts as input a linear programming solution file, packs it in a special data structure and interactively extracts specific information from that file through a set of Queries.

The size of memory which is required to run the system depends on the size of the file to be accommodated, and thus on the size of the original optimization problem. The user extracts information from the solution file with a Query Language, asking questions related to the solution of the problem represented in the file.

The entire system has been developed in FORTRAN language for portability and better coordination with other Linear Programming procedures which are also written in FORTRAN.

The Queries are self-documenting and their syntax follows closely the syntax of the English language. To avoid typing effort for experienced ATHENA users, *short forms* of Queries are provided. Only the characters comprising the short forms are interpreted by the system, with all subsequent contiguous nonblank characters ignored.

B. QUERY LANGUAGE

The Query Language consists of three subsets of Queries:

1. The Control Queries:

With this subset of Queries the user controls mainly the output of the system, inserting comments, headings, etc.

2. The Command Queries:

With these the user communicates with the solution file and extracts the specific information he needs.

3. The SET Command.

This command qualifies the ATHENA queries to access only a subset of the problem file.

1. Control Queries

- a. VERIFY (Short Form V)

All the following Queries will be displayed with the output. This Control Query is useful when the OFFLINE printer is used for the output instead of the terminal, or when the system is used under Batch Processing; in these cases answers are transmitted to the output device without the corresponding questions if the system is not in VERIFY mode.

- b. NOVERIFY (NOV)

The following Queries do not appear with the output. This Control Query is most frequently used when a terminal is used for all output. The DEFAULT mode of the ATHENA system is NOVERIFY.

- c. PROMPT (P)

The system responds with the prompt:

' INPUT A COMMAND '

whenever it is ready to accept a Query. PROMPT is a DEFAULT mode of the system.

d. NOPROMPT (NOP)

Used to avoid the prompting phrase in the output, especially when the OFFLINE printer or Batch Processing is used.

e. H Any Character String

When the first column of a Query is the letter H, then the character string is printed as is in the output. H is used to insert comments or headings in the output.

f. * Any Character String

When the first column of a Query is the character *, no action takes place. This is considered as a comment and is ignored. * is useful to insert comments and/or headings on the terminal output, but *not* on the OFFLINE printer.

g. END (E)

Used to end the current session.

2. Command Queries

A Command Query consists of several fields. Some fields are required and must always appear in a Command Query and others are optional. Each field is separated from the others by at least one blank character. The number of blanks between fields is not significant and a Query may start at any character position in the command. The length of a Query cannot exceed 80 characters including the spaces between the fields.

The possible fields that can be included in a Command Query are:

- a. TYPE
- b. SELECT
- c. MASK
- d. CONDITION
- e. PRINT OPTION

The fields in a Command Query must appear in the above sequence and the first 3 of them must always appear, with only 2 exceptions. A detailed description follows of each individual field and the way that it may be used.

a. TYPE field

This is the first field of the Query and may start at any character position. This field can be one of the following:

(1) DISPLAY (D)

Used when all the records which meet the requirements of the other fields are to be displayed in the output in the sequence they are encountered starting from the beginning of the solution file.

The portion of each individual record that will be displayed depends on the PRINT OPTION field.

(2) COUNT (C)

Used when only the number of records which meet the requirements of the other fields is desired. COUNT is especially useful immediately preceding a DISPLAY command so the user will know in advance the size of output, avoiding

unpredictably extensive printouts. For this Query the PRINT OPTION is ignored as meaningless.

(3) ADD (A)

Used when some numerical quantities of the qualified records are to be summed. The names of the numeric quantities of each record that will be added are given in the PRINT OPTION field. If no PRINT OPTION appears, all the numeric quantities of each record are added and their sums are displayed with appropriate labels.

Since it is mathematically meaningless to add LOWER or UPPER LIMITS, or PENALTIES, they can not be summed or displayed.

(4) AVERAGE (AV)

Used exactly as the ADD command to display arithmetic averages. The sums are divided by the total number of the qualified records.

(5) ACTIVATE (AC) (Syn. ACTIVE)

With the ACTIVATE command the user can indicate a subset of the records of the solution file with specific qualifications determined by the other fields so that subsequent Queries will implicitly refer only to that subset. The user can expand the initial subset by using the ACTIVE command repeatedly to add new records to the active subset.

The command ACTIVATE can minimize the searching time for the required information in the active subset. Each time the ACTIVATE command is issued, the system

responds with the number of records added to the active subset and the current total number of active records.

(6) DEACTIVATE (DE) (Syn. DEACTIVE)

Used to delete records with specific qualifications from the current active subset - created by the ACTIVE commands - or to eliminate any active file. The system responds with the number of records deactivated and the total number of records remaining active.

The entire active file can be deactivated by:

' DEACTIVE ALL or DE A'.

With this Query all the currently active records will be deactivated and the message:

' F I L E D E A C T I V E '

will be printed out. Subsequent queries will refer to the entire solution file.

b. SELECT field

This field is mandatory and specifies whether the qualified records are ROWS, COLUMNS or BOTH. It may consist of one of the following:

(1) ALL (A)

Specifies that the entire file must be searched for the qualified records starting from the first ROW and continuing to the last COLUMN.

(2) COLUMNS (C)

Specifies that the COLUMNS only will be searched for the qualified records starting with the first COLUMN and continuing to the last COLUMN.

(3) ROWS (R)

Specifies that the ROWS only will be searched for the qualified records starting with the first ROW and continuing to the last ROW.

c. MASK field

Specifies that any record is qualified for processing if the name of the record fits the MASK field. The MASK field is left justified and may contain 1 to 8 characters. All the right unfilled positions up to 8 characters are assumed to be the character *. The MASK is matched against the name, starting from the left, character by character. Any character in the name is matched with a * in the MASK field. The MASK field is mandatory.

EXAMPLES

i. The MASK 'X*****Y' specifies all the names starting with the letter X and having as the 8th (last) character the letter Y.

ii. The MASK 'X' is equivalent with the MASK 'X*****' and means all the names starting with the letter X.

iii. The MASK '*****Y' specifies all the names ending with the letter Y and it is NOT equivalent with the MASK 'Y'.

iv. The MASK 'ABCDEFXY' specifies only this name and since the names of ROWS and COLUMNS are assumed to be inclusively unique, the searching of the file stops when the first match is made.

v. The MASK '*' specifies ALL the names and may be used when no particular mask is desired.

d. CONDITION field

The syntax of this field is:

FOR (<conditional phrase>)

The word FOR, left parenthesis and right parenthesis must always appear when the CONDITION field appears in a Query.

There are two kinds of conditional phrases:

The simple conditional phrase and the compound conditional phrase.

(1) Simple Conditional Phrase

There are 3 kinds of simple conditional phrases: The Relational, the Status and the Bound simple conditional phrases.

(a) Relational Simple Conditional Phrase

The syntax is:

<Arg1> <Relop> <Arg2>

where Arg1, Arg2 and Relop are one of the following:

i. Arg1

X	for	ACTIVITY LEVEL
S or C	for	SLACK ACTIVITY or INPUT COST

L	for	LOWER LIMIT
U	for	UPPER LIMIT
D	for	DUAL ACTIVITY or REDUCED COST
P	for	UPPER PENALTY
W	for	LOWER PENALTY

ii. Relop

Relational operators EQ, NE, GT, GE, LT, LE with the same meaning as in FORTRAN. (Note, however, that there are not imbedded decimal characters as in FORTRAN.)

iii. Arg2

Arg2 is defined exactly as Arg1 with the enhancement that Arg2 may also be any integer or real number. Arg2 cannot be expressed as a floating point number in exponential notation.

(b) Status Simple Conditional Phrase

The syntax is:

STATUS <Flag> or ST <Flag>

where Flag is one of the following:

BS	for	BASIC
LL	for	LOWER LIMT
UL	for	UPPER LIMIT
EQ	for	FIXED
VC	for	VOID COLUMN
SC	for	SINGLETON COLUMN

FC	for	FIXED COLUMN
BC	for	BOUND CHANGED
VR	for	VOID ROW
SR	for	SINGLETON ROW
RR	for	REDUNDANT ROW
FR	for	FREE ROW
ER	for	DOUBLETON EQUATION
TR	for	TIGHTEN RANGE

(c) Bound Simple Conditional Phrase

The syntax is:

<Arg1> MINIMUM or <Arg1> MAXIMUM

where Arg1 is specified as in Relational Simple Conditional Phrase. The words MAXIMUM or MINIMUM can be abbreviated as MAX or MIN, respectively. This is used to extract those records which have the MAXIMUM or MINIMUM value in the specified field with the specified MASK. The system responds with the first record encountered with the maximum or minimum value associated with Arg1, and the total number of records that meet the requirements. This phrase may not be used with ACTIVE or DEACTIVE options in the TYPE field of the Query.

(2) Compound Conditional Phrase

The syntax of this phrase is:

<Relational Cond. Phrase> <Log. Oper.> <Relational Cond. Phrase>

or

<Relational Cond. Phrase> <Log. Oper.> <Status Cond. Phrase>

where Log. Oper. is OR or AND with the meaning of the corresponding logical operators. Note that the Bound Simple Conditional Phrase is not compatible for use in a Compound Conditional Phrase, since it exhibits no boolean value. Also, the Status Conditional Phrase must always appear *after* the logical operator.

The CONDITION field as a field must be separated by at least one blank from the other fields of the Query. The word FOR, the left parenthesis, and the first element of the conditional phrase do not require separation by blank characters, nor do the last element of the conditional phrase and the right parenthesis.

The CONDITION field is optional and need not appear in the Query. If it is not present, any record is qualified if the MASK field is satisfied. Using the ACTIVATE and DEACTIVATE commands the user can actually have unlimited length conditional phrases, by adding qualified subsets of records in the ACTIVE file.

e. PRINT OPTION field

This is the last field of a Query. It is optional, and if it does not appear the entire contents of each record which satisfies both the MASK and the CONDITION fields are printed out.

The elements of the PRINT OPTION field may be any combination of the following:

X, C or S, L, U, D, P, W

with meanings as described in the CONDITION field. The output will include information described in the PRINT OPTION with corresponding headings. The elements of the field can be separated by any number of blanks, by commas, or not at all. If both C and S appear in the PRINT OPTION neither of them is printed out. For the commands ADD and AVERAGE the default PRINT OPTION is X, C or S, D since there is no meaning for LIMITS and PENALTIES.

For all Queries that potentially require more than one output record for the answer (i.e., all Queries except COUNT, ACTIVE, DEACTIVE and SET), the output will include the following entries for each record: NUMBER, NAME, STATUS and the entries specified in the PRINT OPTION field in the sequence in which they appear. At the end of the answer output for each Query the total number of qualified records is given. The heading for the output is determined by the SELECT field. If for this field the option ALL is used, the heading will be the one for ROWS although COLUMNS may also be included in the output.

3. The SET Command

By default each time a Command Query is issued the whole solution file is searched starting at the first ROW or

COLUMN and continuing by examining sequentially all the records.

Queries may sometimes apply only to a small part of the solution file or to records whose relative position in the file is known. In these cases the SET command can cause searching to be initiated at a particular entry in the file and continued to another particular entry. Also a fixed step size can be specified for the search. Thus much computational effort can be avoided.

The syntax for the SET Command is:

```
SET <number 1> <number 2> <number 3>
```

where;

number 1 is the number of the starting record;

number 2 is the number of the record to stop
searching;

number 3 is the step for searching.

All these numbers must be integers separated by at least one blank and the presence of all of them is required. These numbers also must be in the range of total number of records for the file. The SET limits apply to qualify any subsequent search of the file even if ROW or COLUMN subsets are specified by a Query.

EXAMPLE

Suppose the solution file has 300 rows and 2000 columns and the following SET command is issued:

SET 18 1500 10

For all subsequent Queries:

If the SELECT field of the Query is ALL then the searching starts at the 18th record and continues through the 1500th record with step 10 (i.e., Record numbers 18, 28, 38, are examined).

If the SELECT field is ROWS then the searching will start at the 18th row through the last row (300th) with step 10.

If the SELECT field is COLUMNS then the searching will start at 18th column through the 1500th column (or equivalently the 318th record through the 1800th record) since the number of columns is greater than 1500.

To restore default settings, use:

'SET DEFAULT' or 'SET D'

C. ERROR MESSAGES

The following error messages are typed at the terminal as soon as they are detected. If the error is only in syntax, the system is immediately ready to accept a new query, otherwise execution is terminated. Errors have been grouped with one message for each group. Messages are self-explanatory.

ERROR NO

POSSIBLE REASON

1	: Attempt to parse a blank query.
101	: Invalid TYPE field. One of the characters D,V,C or blank was expected after A.

102 : Invalid TYPE field. One of the characters
O,E was expected after S.

103 : Invalid TYPE field. One of the characters
P,V was expected after NO.

104 : Invalid TYPE field. No command starts
with the given letter.

201 : Missing character or somewhere in the
query there is no space delimiter.

202 : There is no space delimiter.

301 : Invalid SELECT field. SELECT field is
missing or there is no space delimiter
between TYPE and SELECT fields.

502 : Invalid CONDITION field. the word FOR is
missing (the string OR was expected after
F), or invalid PRINT field.

503 : Missing left parenthesis in CONDITION field.

504 : Incomplete condition field or missing
space delimiter.

505 : Missing right parenthesis in condition field.

506 : Invalid OR logical operator. Character R
R was expected after O.

507 : Invalid AND logical operator. The string
ND was expected after A.

508 : Invalid logical operator. Only OR and
AND are accepted.

511 : Invalid operand for status. The character
C or S was expected after B.

512 : Invalid operand for status. The character
L was expected after L.

513 : Invalid operand for status. The character
L was expected after U.

514 : Invalid operand for status. The character
Q or R was expected after E.

515 : Invalid operand for status. The character
V, S, F or B was expected before C.

516 : Invalid operand for status. The character
V, S, R, F, E or T was expected before R.

517 : Invalid operand for status. The character
P was expected before P.

518 : Invalid operand for status. The character
I, A, C or D was expected before E.

519 : Missing space delimiter after status
operand.

520 : Non recognizable operand for status.

601 : Invalid first operand for relational
operator in condition field.

602 : Missing space delimiter in a simple
conditional phrase.

603 : Invalid relational operator. The character
T or E was expected after G.

604 : Invalid relational operator. The character
T or E was expected after L.

605 : Invalid relational operator. The character
Q was expected after E.

- 606 : Invalid relational operator. The character
E was expected after N.
- 607 : Invalid operand in bound conditional
phrase. The character N was expected
after string MI.
- 608 : Invalid operand in bound conditional
phrase. The string AX was expected after M.
- 609 : Unrecognizable relational operator in
simple conditional phrase.
- 701 : Invalid print field, or missing word FOR
in condition field.
- 1001 : Error in input data. Unrecognizable
status code.
- 1002 : Error in input data. Data encountered
has less than the expected number of rows
and columns.

D. LIMITATIONS - EXTENSIONS

As mentioned in the introduction, ATHENA is a direct expansion of the PERUSE system. It includes all the features of PERUSE, except the weighted average command, and has the following differences and extensions:

1. ATHENA supports two distinct data structures, each different from that of PERUSE. This was necessary in order to support efficient access to individual records or group of records. The SUPERPARSE data structure is unique to ATHENA.

2. ATHENA accepts as input a simple file which can be easily obtained from the solution file of any linear programming package on tape, disk or cards.

3. ATHENA supports the commands SET, ACTIVATE, DEACTIVATE and COUNT, in addition to the commands of PERUSE, allowing the user to construct logical subsets of the solution and efficiently access these subsets as independent files with very small access time.

4. ATHENA supports compound conditional phrases for extraction of more specific information and the bound conditional phrase for maximum and minimum values.

5. ATHENA uses object time variable format allowing better appearance of output and uses the words NONE and INFINITY instead of the number 0.7273E76 for better readability.

6. ATHENA accepts reduced problems from PREP [3] and can be used to pass the PREP status file with the solution file of any optimization system to permit recovery of the original problem solution.

ATHENA has been designed to handle solution files with up to 30,000 records. The actual limit is imposed by the number of real number values that must be stored explicitly. This number cannot presently exceed 65536 since this is the largest integer pointer value which can be stored by ATHENA in a 2-byte halfword. Experience has shown that the average number of stored values for each record, excluding penalties

is 1.5 [9]. Adding to this another 0.5 per record for penalties, the problem size limit may be as large as 30,000 rows and columns.

A rough estimation of the space needed for the packed file in 4-byte words can be obtained by multiplying the sum of rows and columns of the solution file by 5 for the SPARSE data structure and by 4 for SUPERSPARSE. Before using ATHENA, adjust the size of the SOLFIL array in common block SOLPAC to this number. To avoid passing problems with common areas under some time-sharing systems, use an array size which is an exact multiple of the intrinsic page size, and which is larger than the number calculated above. Also make corresponding adjustments to the DEFINE FILE statement of the main program (e.g. use multiples of 4,096 for IBM systems).

ATHENA has been developed in modular form and can be easily changed or extended to support future needs. Commands which can be easily implemented include the weighted average, the sort of output, or further calculations needed for the analysis of the solution file. ATHENA can also be used as part of an integrated system for sensitivity analysis of optimization problems.

E. SYSTEM INTERFACE

1. Preview

Linear Programming packages give differing forms of output so that it is difficult for a system to be interfaced

with all of these solution formats. ATHENA accepts as input a solution file in a 'standard' form which can be easily obtained from any other solution file form.

ATHENA is best utilized in an interactive system, although it can also be used in batch processing. On the other hand, most L.P. packages run only in batch processing. An exception is XS [4]. Moreover, in some systems there is no integration of interactive and batch processing. In these cases, the solution files may be transferred manually from one system to the other using magnetic tapes or cards.

A simple input file has been designed which can be punched in cards or entered on tape, disk, or other storage media.

2. Input File

The input file consists of records with the following structure:

a. The first record always contains the number of ROWS, the number of COLUMNS, and in position 51 the character '1' if each ROW record contains PENALTIES or '0' otherwise. the FORMAT of the first record is

(I5,30X,I5,10X,I1).

b. Each subsequent record contains explicitly all the information associated with each ROW and COLUMN, with the following format:

NAME Format 2A4 (left justified)
STATUS Format A2

X, C or S, L, U, and D numeric field values with Format 5E14.5 or 5F14.5. (The meanings of each of these fields is described in the previous section.) If the solution file includes PENALTIES, then two records will be associated with each ROW. The first will be exactly that described above, and the second will have the FORMAT(E14.5,16X,E14.5) for P, and W, the UPPER and LOWER PENALTIES. In all cases, INFINITE values will be represented explicitly by the number $\pm 0.1E76$. The total number of records must agree with the sum of ROWS plus COLUMNS, with the records of ROWS preceding those of COLUMNS; otherwise an Input error will occur. The file is read in and packed one record at a time.

3. Packed File

a. Packed File as Input

If the input file is already packed from a previous use of the system, it will be read in unformatted binary form. The system will provide the user information for memory requirements before reading the file. The packed file may be on a tape or disk but cannot be on cards. The system will ask the user at the beginning of a session for the number of the file.

b. Packed File as Output

If an unpacked file is used as input, the system will ask for the file number where a packed file is to be

written. Of course, a DEFINE FILE statement must be included right after the declarations of the main program.

E. EXAMPLE OF SYSTEM USE

The procedure follows for use of ATHENA at the Naval Postgraduate School Computer Center with the IBM 360/67. The solution file here is produced by the MPS/360 package and ATHENA is used under CP/CMS. Similar procedures can be followed for any other installation.

1. Obtaining the Unpacked Solution

a. Submit the problem to be solved using the usual Control Cards required for the MPS/360 package inserting before the Control Card:

```
//MPS2.SYSIN DD *
```

the following cards:

```
//MPS2.SYSPRINT DD DSN=Sxxxx.nnnnnn,  
// UNIT=3330,VOL=SER=DISK04,  
// SPACE=(CYL,(1,1)),DISP=(NEW,KEEP),  
// DCB=(RECFM=UA,BLKSIZE=133)
```

With these cards the output of MPS will go to the DISK instead of the printer.

xxxx is the user's number and
nnnnnn is the file name on the disk.

If the solution file is too big or disk space is not available use tape or tapes to store the output.

b. Use the program REWRITE (see [2]) to transform the MPS/360 tape or cards to the format required for the ATHENA unpacked Input file.

c. Now the unpacked file for ATHENA is available and can be used to analyze the solution.

2. Using the System Under CP/CMS

ATHENA in CP/CMS TEXT form requires about 67K bytes. The space needed for the packed file depends on the number of records, the method used for packing and the density of the original file. 200K bytes would be sufficient to hold a packed file with up to 13,000 rows and columns. After sufficient space has been secured, the following procedure may be applied:

a. Ask OPERATOR to connect the tape with the Input file created by the REWRITE to the private disk as device 181. As soon as the tape is connected, the message 'DEVICE 181 ATTACHED' will be printed at the terminal.

b. Before using the tape, type ALWAYS under CMS the command ' TAPE SKIP 1 '. This command will position the tape at the first record of the file. This command is required because the tape created by IBM O.S./360.

c. Type \$ ATHENA

ATHENA will ask for information about the file identifiers for input-output, whether the file is packed and method of packing and will give the size of the packed

file. For input file ordinal use any number between 01 and 99 excluding the numbers 03 - 06. For output file ordinal give the number 03 or 04. These are the numbers used by the DEFINE FILE FORTRAN statement and they can be changed. The system will be ready to accept QUERIES as soon as the prompt phrase 'INPUT A COMMAND' is typed by ATHENA at the terminal. The packed file can be saved on a tape using the 'TAPE DUMP' command under CMS.

In the next few pages a demonstration of using ATHENA with a solution file of 766 rows and 10921 columns is given. This problem is a mixed integer optimization model with 963 binary variables for medium term capital budgeting of the Naval Air Test Center [8]. For this problem, a query may require as much as one and a half minutes of clock time if the interactive system on the IBM 360/67 is under heavy use and the query is difficult to answer. However, most queries are answered almost immediately. Response time is especially good when the user makes use of ACTIVATE, MASK and SET features to qualify necessary searching.


```

506 ROWS OR COLUMNS WITH MASK : ****y****
SATISFY THE CONDITIONS
*
* HOW MANY OF THEM ARE EQUAL ZERO ?
*
C C * FOR( X EQ 0, AND ST BS)

55 ROWS OR COLUMNS WITH MASK : ****x****
SATISFY THE CONDITIONS
*
* USING THE COMMAND 'ACTIVATE' ONLY THE ACTIVATED
* RECORDS ARE SEARCHED TO ANSWER THE QUESTION.
* THIS IS A GOOD WAY TO AVOID SEARCH OF THE WHOLE
* FILE.
*
ACTIVATE COLUMNS X0

73 RECORDS ACTIVATED
TOTAL RECORDS ACTIVE : 73
*
* HOW MANY ROWS NOW ?
*
C R *

0 ROWS OR COLUMNS WITH MASK : ****x****
SATISFY THE CONDITIONS
*
* THERE ARE NO ROWS SINCE ONLY COLUMNS ACTIVATED.
* HOW MANY OF THEM ARE BASIC ?
*
C C * FOR(ST BS)

1 ROWS OR COLUMNS WITH MASK : ****x****
SATISFY THE CONDITIONS
*
* DISPLAY THEM
*
DISPLAY C * FOR(ST BS) X L
THE FOLLOWING ROWS OR COLUMNS SATISFY CONDITIONS
NUMBER .NAME.. AT ...ACTIVITY... ..LOWER LIMIT.
803 X041 BS 0.12910 0.00000

1 ROWS OR COLUMNS WITH MASK : ****x****
SATISFY THE CONDITIONS
*
* WHAT IS THE AVERAGE OF NONZERO ACTIVITIES ?
*
AVERAGE C * FOR(X NE 0)
THE FOLLOWING ROWS OR COLUMNS SATISFY CONDITIONS

```

NUMBER .NAME.. AT ...ACTIVITY... ..INPUT COST..
SUMS OR AVERAGES : 0.98773 0.76576

71 ROWS OR COLUMNS WITH MASK : *****
SATISFY THE CONDITIONS

*
* THE SUM OF ACTIVITIES AND INPUT COST ?
*

ADD C * FOR (X NE 0)
THE FOLLOWING ROWS OR COLUMNS SATISFY CONDITIONS

NUMBER .NAME.. AT ...ACTIVITY... ..INPUT COST..
SUMS OR AVERAGES : 70.12909 54.36832

71 ROWS OR COLUMNS WITH MASK : *****

SATISFY THE CONDITIONS

*
* ADD SOME NEW COLUMNS AT THE ACTIVE FILE
*
ACT C YM01

175 RECORDS ACTIVATED
TOTAL RECORDS ACTIVE : 248

*
* BASIC ?
*
C C * FOR (ST BS)

7 ROWS OR COLUMNS WITH MASK : *****
SATISFY THE CONDITIONS

*
* DISPLAY BASIC AND ZERO ACTIVITY
*

D C * FOR (X EQ 0 AND STATUS BS) X C
THE FOLLOWING ROWS OR COLUMNS SATISFY CONDITIONS

NUMBER .NAME.. AT ...ACTIVITY... ..INPUT COST..
0068 YM01061 BS 0.00000 -0.01000

1 ROWS OR COLUMNS WITH MASK : *****
SATISFY THE CONDITIONS

*
* HOW MANY INPUT COSTS ARE ZERO ?
*
C C * FOR(C EQ 0)

5 ROWS OR COLUMNS WITH MASK : *****
SATISFY THE CONDITIONS

*
* HOW MANY AT LOWER LIMIT ?
*

C C Y FOR(STATUS LL)

171 ROWS OR COLUMNS WITH MASK : *****
SATISFY THE CONDITIONS

*
* ELIMINATE SOME RECORDS FROM THE ACTIVE FILE
*

DEACTIVATE C X

73 RECORDS DEACTIVATED
TOTAL RECORDS ACTIVE : 175

*
* CHECK FOR THE ELIMINATION
*

C C X

0 ROWS OR COLUMNS WITH MASK : *****
SATISFY THE CONDITIONS

*
* ELIMINATION O.K.
*

* HOW MANY AT LOWER LIMIT (LL) NOW ?

*
C C * FOR (ST LL)

169 ROWS OR COLUMNS WITH MASK : *****
SATISFY THE CONDITIONS

*
* WHAT IS THE AVERAGE ?
*

AV C Y FOR(ST LL)
THE FOLLOWING ROWS OR COLUMNS SATISFY CONDITIONS

NUMBER	NAME..	AT	..ACTIVITY..	..INPUT COST..
SUMS OR AVERAGES :			0.00000	-0.01000

169 ROWS OR COLUMNS WITH MASK : *****
SATISFY THE CONDITIONS

*
* RETURN TO THE ORIGINAL BIG FILE
*

DEACTIVATE ALL
F I L E D E A C T I V E

*
 * DISPLAY THE 5 FIRST ROWS
 *

SET I 5 1
 U R * X S
 THE FOLLOWING ROWS OR COLUMNS SATISFY CONDITIONS

NUMBER	NAME..	AT	...ACTIVITY...	SLACK ACTIVITY
1	BENEFITS	BS	449.49902	-449.49902
2	RC1011	UL	0.00000	0.00000
3	RC1021	UL	0.00000	0.00000
4	RC1031	UL	0.00000	0.00000
5	RC1041	BS	0.00000	0.00000

5 ROWS OR COLUMNS WITH MASK : *****
 SATISFY THE CONDITIONS

*
 * THE FIRST 5 COLUMNS
 *

D C * C D
 THE FOLLOWING ROWS OR COLUMNS SATISFY CONDITIONS

NUMBER	NAME..	AT	..INPUT COST..	..REDUCED COST.
757	X001	UL	0.99994	0.33312
768	X002	UL	0.89108	0.77450
769	X003	UL	0.45609	0.23338
770	X004	UL	0.45483	0.36919
771	X005	UL	0.73440	0.60813

5 ROWS OR COLUMNS WITH MASK : *****
 SATISFY THE CONDITIONS

*
 * RETURN TO THE ORIGINAL FILE
 *

SET DEFAULT

*
 * DISPLAY VARIABLES IN A SPECIFIC RANGE
 *

D C YM FOR (X GT 10.5 AND X LT 25.5) X L
 THE FOLLOWING ROWS OR COLUMNS SATISFY CONDITIONS

NUMBER	NAME..	AT	...ACTIVITY...	..LOWER LIMIT.
5708	YM05061	BS	14.00000	0.00000
6139	YM17171	BS	11.54310	0.00000
6247	YM20201	BS	17.13029	0.00000
6758	YM35061	BS	17.29999	0.00000
7472	YM20202	BS	18.98279	0.00000
7983	YM35062	BS	11.90000	0.00000
8697	YM20203	BS	21.00000	0.00000
9208	YM35063	BS	16.59999	0.00000
9378	YM05014	BS	11.90000	0.00000
9922	YM20204	BS	13.00000	0.00000
10133	YM35064	BS	23.06062	0.00000
10608	YM05065	BS	14.00000	0.00000
11147	YM20205	BS	13.00000	0.00000
11658	YM35065	BS	19.50000	0.00000

14 ROWS OR COLUMNS WITH MASK : YM*****
SATISFY THE CONDITIONS

*
? WHAT IS THE MAXIMUM OF THE ACTIVITY LEVEL ?
*

D C * FOR(X MAX) X C
1 YM13194 BS 83.63869 0.00000

1 ROWS OR COLUMNS WITH MASK : Y*****
SATISFY THE CONDITIONS

*
* THE MINIMUM LOWER LIMIT FOR THE ROWS ?
*

D R * FOR(L MIN) L U
766 BENEFITS BS NONE NONE

736 ROWS OR COLUMNS WITH MASK : Y*****
SATISFY THE CONDITIONS

*
? THE MAXIMUM ACTIVITY LEVEL FROM VARIABLES
* STARTING WITH LETTER Y ?
*

D C Y FOR (X MAX) X C
1 YM18184 BS 88.63869 0.00000

1 ROWS OR COLUMNS WITH MASK : Y*****
SATISFY THE CONDITIONS

*
? ARE THERE INFEASIBILITIES ?
*
COUNT ALL * FOR (ST IN)

0 ROWS OR COLUMNS WITH MASK : *****

SATISFY THE CONDITIONS

*
? EXIT FROM THE SYSTEM
*

END

Acknowledgment

ATHENA has been chosen for the name of this system for several reasons. First, ATHENA is the goddess of wisdom from ancient Greece. Also, it is the Greek name of the capitol of the country of one of the authors, as well as his mother's name.

References

1. Aho, A.V. and Ullman, J.D., Principles of Compiler Design, Addison-Wesley, 1977.
2. Bradley, G.H., Brown, G.G. and Galatas, P.I., "ATHENA: An Interactive System to Analyze Large-Scale Optimization Models," NPS52-80-005, April 1980, Naval Postgraduate School Technical Report.
3. Bradley, G.H. Brown, G.G. and Graves, G.W., "Preprocessing Large-Scale Optimization Models," ORSA/TIMS, Atlanta, November 1977.
4. Brown, G.G. and Graves, G.W., "Design and Implementation of Large-Scale (Mixed Integer) Optimization System," ORSA/TIMS, Las Vegas, November 1975.
5. Galatas, P.I., "ATHENA: A System to Interactively Analyze Large-Scale Optimization Models," M.S. Thesis, Naval Postgraduate School, March 1979.
6. International Business Machines, Mathematical Programming System/360; Version 2, Linear and Separable Programming--User's Manual, 1971.
7. International Business Machines, Control Program 67/Cambridge Monitor System CP/CMS, Version 3.25, 1974
8. Mavrikas, C., "Optimal 5-year Planning Using Mixed-Integer Linear Programming--Three Models Implemented for Naval Air Test Center," M.S. Thesis, Naval Postgraduate School, September 1979.
9. O'Neil, R.P. and Sanders, R.C., "PERUSE System Manual--Version 2," September 1977.

INITIAL DISTRIBUTION LIST

	<u>No. Copies</u>
1. Office of Naval Research Code 434 800 North Quincy Street Arlington, VA 22217	2
2. R. Stampfel Code 55 Naval Postgraduate School Monterey, CA 93940	2
3. Defense Documentation Center Cameron Station Alexandria, VA 22314	2
4. Library, Code 0142 Naval Postgraduate School Monterey, CA 93940	2
5. Chairman, Code 55 Department of Operations Research Naval Postgraduate School Monterey, CA 93940	2
6. Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, CA 93940	2
7. Professor Gordon Bradley, Code 52Bz Naval Postgraduate School Monterey, CA 93940	10
8. Professor Gerald Brown, Code 55Bw Naval Postgraduate School Monterey, CA 93940	20
9. Captain Panagiotis I. Galatas Alketou 19, Pagrati Athens 506, Greece	3
10. Professor Glenn W. Graves Graduate School of Management University of California Los Angeles, CA 90024	1

END

DATE
FILMED

3-81

DTIC