AD-A048 299 TEXAS UNIV AT AUSTIN CENTER FOR CYBERNETIC STUDIES F/G 9/2
AN EVALUATION OF MATHEMATICAL PROGRAMMING AND MINICOMPUTERS.(U)
OCT 77 J ELAM, D KLINGMAN, J MULVEY N00014-76-C-0616
UNCLASSIFIED CCS-313 NL

1 of 1
ADA048 299

END
DATE FILMED
2-78
DDC

12
NW

CENTER FOR CYBERNETIC STUDIES

The University of Texas
Austin, Texas 78712

see back page for 1473

Research Report CCS 313

AN EVALUATION OF MATHEMATICAL PROGRAMMING AND MINICOMPUTERS

by

Joyce Elam*
Darwin Klingman**
John Mulvey***

October 1977

* Department of Decision Sciences, The Wharton School, University of Pennsylvania, Philadelphia, PA 19104

** Department of General Business, The University of Texas at Austin, Austin, TX 78712

*** Graduate School of Business Administration, Harvard University, Boston, MA 02163

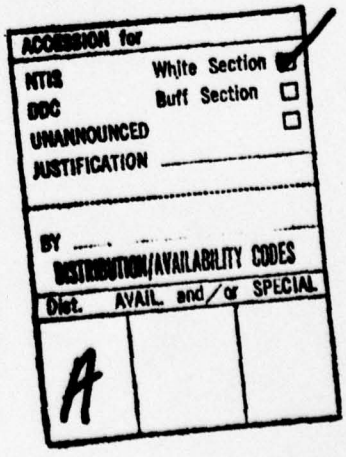
This research was partly supported by ONR Contract N00014-76-C-0383 with Decision Analysis and Research Institute, Project NR047-021, ONR Contracts N00014-75-C-0616 and N00014-75-C-0569 and Department of Transportation Contract DOT-OS-70074 with the Center for Cybernetic Studies, The University of Texas. Reproduction in whole or in part is permitted for any purpose of the United States Government.

CENTER FOR CYBERNETIC STUDIES

A. Charnes, Director
Business-Economics Building, 203E
The University of Texas at Austin
Austin, Texas 78712
(512) 471-1821

ABSTRACT

The purpose of this paper is to explore the viability of implementing mathematical programming algorithms on minicomputers. We feel that the minicomputer explosion presents many exciting research challenges in terms of both identifying and matching appropriate algorithms and applications for such hardware. This paper partially examines these issues by comparing the computational performance of two shortest path algorithms on four very distinct types of computer hardware.



AN EVALUATION OF MATHEMATICAL PROGRAMMING AND MINICOMPUTERS

1.0. Introduction

The purpose of this paper is to explore the viability of implementing mathematical programming algorithms on minicomputers. We feel that the minicomputer explosion presents many exciting research challenges in terms of both identifying and matching appropriate algorithms and applications for such hardware. This paper partially examines these issues by comparing the computational performance of two shortest path algorithms on four very distinct types of computer hardware.

Today, minicomputers represent the single largest category of installed and working computers. Over 200,000 units are operational and the current growth rate is over 30% per year [21]. The remarkable growth of the minicomputer industry in the past decade can be attributed to innovations in technology which have steadily increased the capabilities of minicomputers while decreasing the cost in an equally rapid fashion. Many organizations who previously found computers too expensive are finding that minicomputers offer low cost systems that are simple and easy to use.

The availability of efficient mathematical programming software on minicomputer systems could greatly increase the use of operations research techniques in industry and government simply by placing a decision-making capability at the fingertips of managers. In fact, since minicomputers are relatively inexpensive, it is also possible to dedicate a minicomputer to a specific decision problem and thus create an environment which is conducive to an effective man/machine decision-making system. This could be accomplished by designing the operating system of the minicomputer to improve

the human engineering aspects of problem solving -- that is, to minimize the difficulties of entering, modifying and verifying problem data, passing the problem data to the solution procedure, interpreting the output, and so forth. In addition, the availability of optimization software would make it possible to demonstrate mathematical programming techniques easily and conveniently to managers in their own offices, and at management seminars. Thus incorporating mathematical programming techniques into available minicomputer software is clearly desirable. However, the feasibility and implications of accomplishing this are currently unknown.

This report is arranged as follows. Section 2 of this paper describes the experimental design (the computer codes, computers, test problems, and evaluation criteria) used in the computational study comparing the shortest path algorithms. Section 3 summarizes the empirical results of the study and indicates conclusions about what types of mathematical programming techniques are computationally attractive for a minicomputer. The final section indicates some areas where additional research is needed.

2.0. Experimental Design

2.1. Computer Codes

Among the simplest mathematical programs are shortest path (SP) algorithms, i.e., finding the paths of least distance from a single node (called the root) to the remaining nodes in a directed network. Such algorithms require a minimal amount of storage, can be coded in approximately one hundred BASIC or FORTRAN statements, and have been extensively analyzed. For these reasons, SP methods were chosen to be tested in this study as candidates for implementation on a minicomputer.

Methods for solving shortest path problems fall into two general categories -- label-correcting and label-setting [7, 10]. Based on the computational studies of [7, 10], we choose to use the labeling-correcting method proposed by Pape [18] and the label-setting method proposed by Dial [6]. The characteristics of these codes are shown in Table I.

Both codes were initially obtained in the form of FORTRAN IV source listings [7], and each was translated directly into BASIC. BASIC was chosen since it has become the de facto standard for minicomputers. It is interesting to note that, with the exception of the statements used to assign an external file for input and to call the system internal clock for timing, identical BASIC codes were used on all four computers.

2.2. Computer Systems

Four computers were used in this study -- two minicomputers (DATAPOINT 5500, DATA GENERAL NOVA 840), one large general-purpose computer (CDC 6600), and one medium-size computer (DEC-10). The term "minicomputer" is not well defined in the literature. It is associated with computers which have a wide variety of capabilities and characteristics. The DATAPOINT and the NOVA computers were chosen as being typical and representative of the extremes in the spectrum of minicomputers. The major characteristics of the actual machines used in this study are summarized in Table II.

The CDC 6600 operates under the UT-2D operating system (a specialized operating system written by the University of Texas Computation Center) and has 131,000 60-bit words of central memory. The DEC-10 operates under the Time-sharing Operating System (TOPS-10) and has 256,000 36-bit words of central memory. A full range of peripherals and an extensive software library, including a BASIC compiler, is available on both the CDC 6600 and the DEC-10 systems.

Table I

Computer Code Characteristics

CODE NAME	PAPE	DIAL
ALGORITHM CLASS	LABEL-CORRECTING	LABEL SETTING
NUMBER OF LINES OF SOURCE CODE	91	124
ARRAY STORAGE REQUIRED	4 NODE LENGTH 2 ARC LENGTH	6 NODE LENGTH 2 ARC LENGTH 1 MAXIMUM DISTANCE LENGTH

TABLE II

Computer System Characteristics

	CDC 6600	DEC-10	NOVA 830	DATAPOINT 5500
MAIN STORAGE TYPE	CORE	CORE	CORE	MOS
WORD LENGTH	60 BITS	36 BITS	16 BITS	8-BIT BYTE
CHARACTER LENGTH, BITS	6	7	8	8
CYCLE TIME, MICROSECONDS	1/MAJOR .1/MINOR	.95	1.2	1.6
OPERATING SYSTEM	UT-2D	TOPS-10	RDOS	DOS
MEMORY SIZE	131K WORDS	256K WORDS	64K WORDS	64K BYTES
PERIPHERALS: DISKS	FULL RANGE OF PERIPHERALS	FULL RANGE OF PERIPHERALS	4 CARTRIDGES 5 MEGABYTES EACH	2 PACKS 25 MEGABYTES EACH
TAPE DRIVERS			1 8-TRACK	2 CASSETTES
PRINTERS			1 LINE PRINTER	1 LINE PRINTER
CARD READERS			NONE	1
TYPE OF BASIC	COMPILER	COMPILER	INTERPRETER	INTERPRETER
APPROXIMATE SYSTEM COST	$6M	$600K	$50K	$40K

The NOVA 830 operates under the Real-Time Disk Operating System (RDOS) and has 64,000 16-bit words of central memory. The DATAPOINT 5500 operates under the Disk Operating System (DOS) and has 64,000 8-bit bytes of addressable central memory. The NOVA 830 has 4 disk cartridges, each with 5 megabytes of storage; 1 8-track tape drive; 1 line printer; and several CRT and teletype terminals. The DATAPOINT 5500 has 2 disk packs, each with 25 megabytes of storage; 2 cassette tape readers; 1 line printer; and 1 card reader. The software libraries of both systems contain a BASIC interpreter. In BASIC, each number is represented as floating point which requires 2 words of storage on the NOVA and 4 bytes of storage on the DATAPOINT 5500.

The cost and cycle time of these computers vary greatly. The approximate costs of these computer systems (CDC 6600, DEC-10, NOVA 830, and DATAPOINT 5500) are $6,000,000, $600,000, $50,000, and $40,000 respectively. The cycle times for the DEC-10, NOVA 830, and DATAPOINT 5500 are .95, 1.2, and 1.6 microseconds, respectively. (Cycle time is the minimum time interval that elapses between the start of two successive accesses to any one storage location.) Due to the special architecture of the CDC 6600, both a major and a minor cycle time are defined for this system. In particular, the CDC 6600 central processing unit contains a stack which holds eight words from main storage. The minimum time that must elapse between the starts of two successive accesses to the stack is a minor cycle. The minimum time that must elapse between the starts of two successive accesses to central memory, which includes the time required to fill the stack, is a major cycle. The minor and major cycle times for the CDC 6600 are 1 and 1. microseconds, respectively

Cycle time is often used as a performance indicator of computer systems whereby inference on how a code will perform on another computer are

based on cycle time ratios. We shall see that cycle time is an unreliable measure for the relative performance of shortest path techniques.

2.3. Test Problem

A variety of randomly-generated shortest path problems were used. The problems were created on the CDC 6600 using two FORTRAN program generators. All testing was conducted with the identical test problems.

The shortest path problems which were generated fell into two distinct categories -- grid problems (which are typical of highway networks), and random problems (which are typical of critical path problems). Structurally, a grid problem can be represented by a rectangular matrix such that each matrix position is represented by a node. The only permissible arcs lie between adjacent nodes. In contrast, the adjacent node requirement is dropped in a random network.

The grid network test problems consisted of 25 node and 100 node networks with rectangularities of 5 x 5, 10 x 10, 5 x 20, and 2 x 50. A uniform probability distribution was used with two ranges for the lengths of the distances between nodes, i.e., [1,200] and [1,800].

The random network test problems contained 100 nodes with an average of 5, 10, 15, and 20 arcs per node. The nodes which form endpoints for the arcs in the network were also selected using a uniform probability distribution, subject to the restrictions that an arc cannot originate and terminate at the same node and that duplicate arcs are not allowed. For each network size, two problems were generated, one with arc lengths [1,200] and the other with arc lengths [1,800]. Again, the arc lengths were selected by means of a uniform probability distribution.

The problem specifications are shown in Table III. These problems represent a scaled-down subset of the 2500-node grid network test problems and the 1000-node random network test problems used in the study by Dial, et al. [7]. The test problems in this study have fewer nodes than the larger network test problems in [7] but have approximately the same average number of arcs per node. For the grid problems, the same degree of rectangularity is also maintained. These networks were chosen because larger problems could not be stored in the memory of the DATAPOINT 5500 computer.

2.4. Performance Criterion

We employed central processing time as a measurement of each code's performance. For many applications, such as real-time routing and scheduling of vehicles, the solution time is an important factor which limits the use-ability of an algorithm. This is especially pertinent for minicomputers because of their relatively slow processing speeds. Also, we do not foresee the useage of multiprogramming operating systems on minicomputers; hence, central processing time could be used without introducing random errors in the data.

2.5. Other Experimental Design Considerations

In all machines except the DATAPOINT 5500, internal clock routines were employed for timing computation. Since the DATAPOINT 5500 does not have a user accessible internal clock, timing was performed with a stopwatch and a BEEP instruction; however, the accuracy was not significantly affected since solution times were generally measured in minutes on the DATAPOINT computer. The reported solution times do not include input or output time. Both codes were written to accept problem specifications in forward star format [7] and neither required any preprocessing prior to optimization. The algorithm initialization steps in which the arrays are set to zero were included, however.

GRID NETWORKS

Problem No.	Nodes	Grid Dimensions Node x Node	Number Of Arcs	Cost Range
1.	25	5 x 5	80	1 - 200
2.	100	10 x 10	360	1 - 200
3.	100	5 x 20	350	1 - 200
4.	100	2 x 50	296	1 - 200
5.	25	5 x 5	80	1 - 800
6.	100	10 x 10	360	1 - 800
7.	100	5 x 20	350	1 - 800
8.	100	2 x 50	296	1 - 800

RANDOM NETWORKS

Problem No.	Nodes	Average Arcs per Node	Number Of Arcs	Cost Range
9.	100	5	532	1 - 200
10.	100	10	949	1 - 200
11.	100	15	1562	1 - 200
12.	100	20	1947	1 - 200
13.	100	5	532	1 - 800
14.	100	10	949	1 - 800
15.	100	15	1562	1 - 800
16.	100	20	1947	1 - 800

Table III

Test Problem Specifications

In order to reduce the effects that a particular root node might have on solution time, each of the 16 networks was solved five times (i.e., for five different roots) on each machine (with the exception of problems 13 and 16 which were not available on the NOVA 830) with each code. Median solution times are used as a basis for comparison.

3.0. Computational Results

Computational results are summarized in Table IV. Notice the magnitudes of difference between solution times on the minicomputers and on the larger computer systems. The differences can be attributed to both slower execution speeds on the minicomputers and to the fact that BASIC is compiled on the larger systems while it is interpreted on the minicomputers. It is interesting to note that the differences in the cycle times between the computer systems in this study are relatively small when compared to the differences in run times, and that a system with a smaller cycle time does not necessarily have a smaller run time (the DEC-10 cycle time is shorter than the CDC 6600 major cycle time). Thus, the overall performance of a computer system not only depends on cycle time, but also on the flexibility and power of its instruction repertoire, the number of storage cycles per execution of an instruction, and the input/output capabilities.

These results strongly indicate that there is consistency in the relative behavior of the SP algorithms across all the computer systems used in this study. In particular, the following observations appear to hold regardless of the computer system on which the algorithms were tested. The distance range has very little effect on the PAPE code and grid rectangularities have negligible effects. In fact, the primary parameter affecting solution times of the PAPE code (other than number of nodes) is the arc density of the network. Conversely,

COMPUTER	CDC 6600		DEC-10		NOVA 830		DATAPOINT 5500	
CODE NAME	PAPE	DIAL	PAPE	DIAL	PAPE	DIAL	PAPE	DIAL
			GRID NETWORKS					
			DISTANCE RANGE [1,200]					
Problem No.								
1	.008	.024	.06	.15	1.5	6.8	12	31
2	.032	.057	.23	.38	7	16.8	49	85
3	.031	.073	.21	.46	7	20.59	49	86
4	.033	.105	.18	.61	6.09	34	43	132
			DISTANCE RANGE [1,800]					
5	.008	.064	.06	.38	1.5	21	11	87
6	.035	.124	.21	.75	7.1	39.39	50	177
7	.031	.182	.23	1.00	6.69	55.8	47	225
8	.03	.305	.2	1.63	5.9	111.1	41	385
			RANDOM NETWORKS					
			DISTANCE RANGE [1,200]					
9	.047	.062	.31	.4	10	16.4	70	89
10	.081	.078	.53	.5	16	20.2	111	115
11	.156	.103	.86	.66	28	27.09	198	155
12	.169	.127	1.01	.8	35	32.2	242	187
			DISTANCE RANGE [1,800]					
13	.055	.105	.36	.6	NA*	NA*	80	145
14	.082	.119	.55	.7	17.39	30.6	118	158
15	.131	.14	.9	.9	28.8	37.9	198	199
16	.164	.157	1.0	.89	NA*	NA*	243	223

NA not available

Table IV

Solution Time Comparison of PAPE and DIAL Codes

Median Time in Seconds

the DIAL code is considerably affected by both the rectangularity of the grid and the distance range. For the grid problems, as the problem becomes more rectangular (i.e., larger arc/node rations) and as the distance range increases, the solution times increase markedly. For the random problems, the solution times for the DIAL code increase as the distance range and/or the arc density increase.

To compare the techniques, the PAPE code easily dominates the DIAL code on grid problems. Yet as the arc density of the random problems increase, the DIAL code times increase at a slower rate than the PAPE times as shown in Figure 1. Thus, as arc density increases in the random networks, we observe that there is a transition in time advantage from the PAPE code to the DIAL code. This transition occurs at higher arc densities as the distance range increases. For example, the transition is approximately at [8-13] arcs per node for the distance range [1,200] and [15-18] arcs per node for the distance [1,800].

The conclusions drawn from this study concerning the efficiency of SP algorithms for networks with a fixed arc/node ratio and distance range are identical to the study by Dial, et al. [7] which concluded that the DIAL code is superior to other label-setting and label-correcting implementations. This seems to indicate that the relative behavior of an SP algorithm for certain classes of network problems is not greatly affected by choice of programming language, computer system, or size of problem.

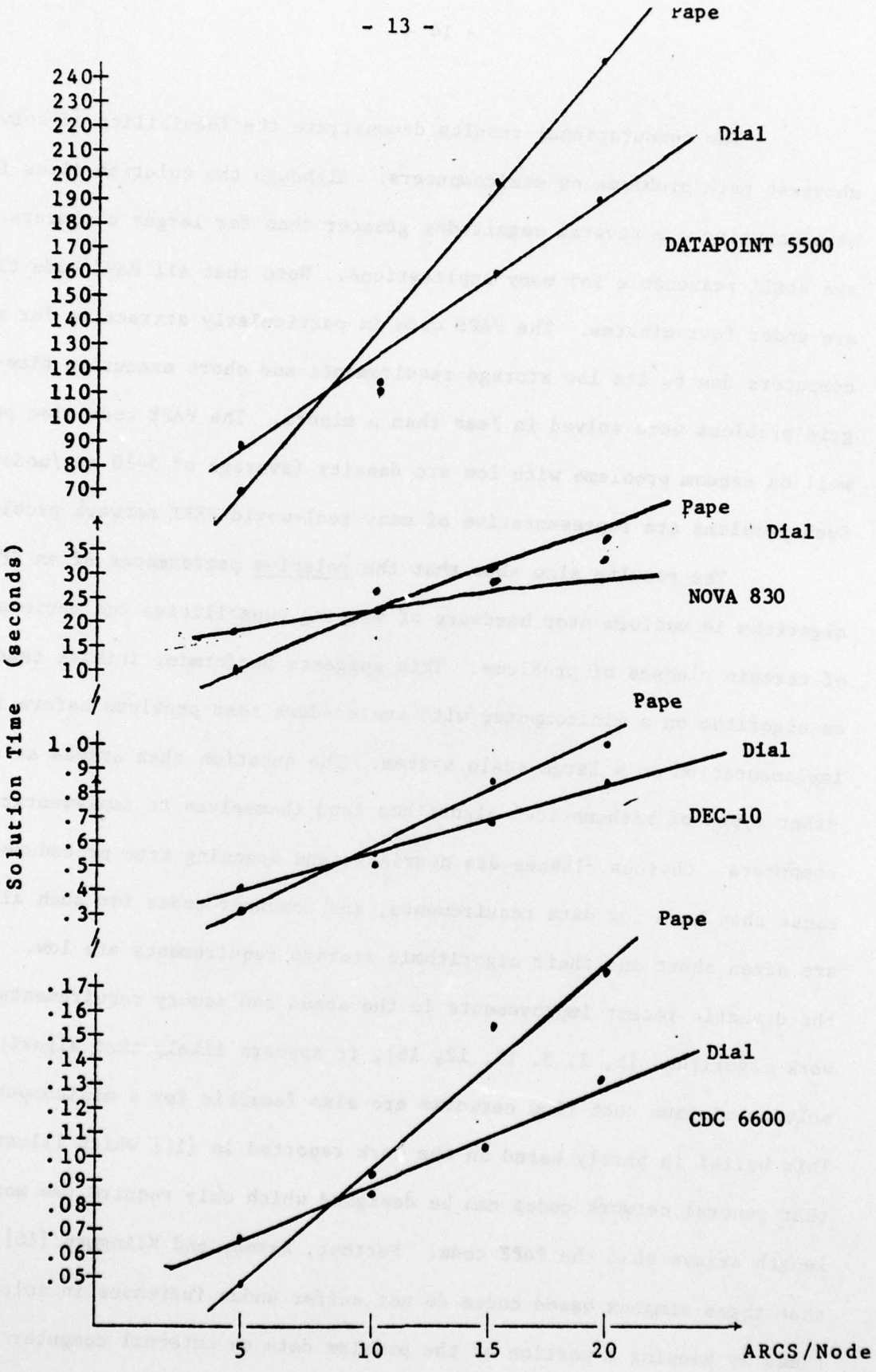


Figure 1

Relationship of Random Network Density to Solution Time for SP Algorithms on Four Computer Systems (Distance Range 1-200)

The computational results demonstrate the feasibility of solving shortest path problems on minicomputers. Although the solution times for minicomputing are several magnitudes greater than for larger computers, they are still reasonable for many applications. Note that all PAPE code times are under four minutes. The PAPE code is particularly attractive for minicomputers due to its low storage requirements and short execution time--all grid problems were solved in less than a minute. The PAPE code also performs well on random problems with low arc density (average of 5-10 arc/nodes). Such problems are representative of many real-world PERT network problems.

The results also show that the relative performance of an SP algorithm is uniform over hardware of varying capabilities and various sizes of certain classes of problems. This suggests performing initial testing of an algorithm on a minicomputer with scaled-down test problems before final implementation on a large-scale system. The question then arises as to what other types of mathematical algorithms lend themselves to implementation on minicomputers. Obvious classes are heuristic and spanning tree procedures because they have low data requirements, and computer codes for such algorithms are often short and their algorithmic storage requirements are low. Given the dramatic recent improvements in the speed and memory requirements of network algorithms [1, 2, 3, 11, 12, 16], it appears likely that algorithms for solving minimum cost flow networks are also feasible for a minicomputer. This belief is partly based on the work reported in [11] which illustrates that general network codes can be designed which only require two more node length arrays than the PAPE code. Further, Karney and Klingman [16] indicate that these simplex based codes do not suffer undue increases in solution times by keeping a portion of the problem data on external computer memory (i.e., on a disk file). Thus, it appears possible to implement an efficient

in-core/out-of-core network code when the problem data cannot be stored within internal memory. Due to the basic similarities between the label-correcting algorithm and the primal simplex minimum cost flow network algorithm, solution times for the network codes can be approximated from the solution times of the PAPE code. The differences in solution times between the two codes for a given network is largely due to the time spent in changing node potentials (dual variable values) [7, 11]. Experience has shown that the node potentials change 1½ times, on the average, for the label-correcting algorithms [7] and 6-10 times for the minimum cost flow network algorithm. Thus, we estimate that the solution times for solving network problems will be roughly 6-10 times longer than the times for solving shortest path problems using the PAPE code.

4.0 FUTURE DIRECTIONS

The emergence of minicomputer network and heuristic codes could substantially increase the use of mathematical programming since many real-world applications are network type problems and the major components of many scheduling systems are network-based [3, 4, 5, 10, 13, 14, 20]. One time-sharing service reports that hundreds of thousands of dollars are spent for PERT analysis alone by their users. Given that problems of this type can be solved on a minicomputer, it would be desirable to design effective, dedicated man/machine systems where the operating system is completely tailored to the application and, perhaps, to the user. Thus, while mathematical programming is at present used primarily by larger companies and governmental agencies, the envisioned coupling of inexpensive minicomputers and network algorithms suggest that it might soon be possible for smaller firms to use them. As mentioned earlier, the capability of a minicomputer to solve network problems combined with its low cost may allow a minicomputer to be dedicated to a particular network application. This suggests research into the design of

software for effective man/machine interfaces and possibly the design of specialized hardware for performing this interaction.

The implementation of general linear, nonlinear, and integer programming algorithms does not appear to be feasible at this time since the data and storage requirements of these algorithms are much larger than for networks and solution times are generally long even when executed on large computer systems. It may, however, be possible to solve some special classes of integer programming problems via reformulation as integer generalized or pure networks.

In summary, the coupling of inexpensive minicomputers and network algorithms appears to be both attractive and viable. Research is needed before these areas can be successfully wed, however. Multifaceted research is required between algorithmic researchers, systems analysts, and computer designers. In general, research is needed to determine the types of algorithmic techniques which are most effective for minicomputers. Such research would probably confirm our belief that linked-list pointer techniques should be used whenever possible to replace arithmetic operations; however, since random access is often slow on current minicomputer hardware, this points out the need for a string manipulation minicomputer which could efficiently handle linked-list pointers.

With regard to future activities, algorithms should be devised and tested which simultaneously use a series of hardwired minicomputers (that is, an array of minicomputers) for performing the steps of an algorithm...For example, network algorithms might be devised where one minicomputer only computes reduced costs, another minicomputer only computes flow changes, and a third minicomputer only updates linked-list pointers. Although this marriage is a long way in the future, its advantages are apparent and its applications

are abundant. For example, since all 0-1 integer programs [14] can be modeled as integer network problems, these developments might allow integer applications to be solved on minicomputers. Additionally, many multi-criteria decision making problems can be successfully modeled and solved as network problems (cf. Ziever et al. [22] and Collins et al. [4]). The paper (Glover and Klingman [13]) discusses several real-world problems which have been modeled and solved as network problems.

Another fundamental research area is on the interface of information and optimization systems. Since the capabilities of minicomputers are greatly restricted, it is crucial to recognize that information systems and optimization models are part of the same decision support facility (see [17]). Both elements must be taken into consideration during design. Unfortunately, the optimization modelers and the information management specialists usually tread over different terrain and rarely understand or speak each other's language. If this situation continues, it will severely retard what we believe is a most promising area for future applications.

REFERENCES

1. R. Barr, F. Glover, and D. Klingman, "Enhancements of Spanning Tree Labeling Procedures for Network Optimization," Research Report CCS 262, Center for Cybernetic Studies, University of Texas at Austin, 1976.

2. G. Bradley, G. Brown, and G. Graves, "Design and Implementation of Large Scale Primal Transshipment Algorithms," Technical Report NPS55BZBW76091, Naval Postgraduate School, Monterey, California, 1976.

3. A. Charnes, F. Glover, D. Karney, D. Klingman and J. Stutz, "Past, Present, and Future Development, Computational Efficiency, and Practical Use of Large-Scale Transportation and Transshipment Computer Codes," Computers and O.R., 2 (1975).

4. M.A. Collins, L. Cooper, and J.L. Kennington, "Solving the Pipe Network Analysis Problem Using Optimization Techniques," Technical Report IEOR 76008, Southern Methodist University, 1976.

5. D. Dantzig, Linear Programming and Extensions, Princeton University Press, Princeton, New Jersey, 1963.

6. R. Dial, "Algorithm 360 Shortest Path Forest with Topological Ordering," Communications of the ACM, 12 (1969), 632-633.

7. R. Dial, F. Glover, D. Karney, and D. Klingman, "A Computational Analysis of Alternative Algorithms and Labeling Techniques for Finding Shortest Path Trees," Research Report CCS 291, Center for Cybernetic Studies, University of Texas at Austin, 1977.

8. E. Dijkstra, "A note on Two Problems in Connexion with Graphs," Numerical Mathematics, 1 (1959), 269-271.

9. S. Dreyfus, "An Appraisal of Some Shortest-Path Algorithms," Operations Research, 17 (1969), 395-412.

10. J Gilsinn and C. Witzgall, "A Performance Comparison of Labeling Algorithms for Calculating Shortest Path Trees," NBS Technical Note 772, U.S. Department of Commerce, 1973.

11. F. Glover, D. Karney, and D. Klingman, "Implementation and Computational Study on Start Procedures and Basis Change Criteria for a Primal Network Code," Networks, 4 (1974), 191-212.

12. F. Glover, D. Karney, D. Klingman, and A. Napier, "A Computational Study on Start Procedures Basis Change Criteria, and Solution Algorithms for Transportation Problems," Management Science, 20 (1974), 793-813.

13. F. Glover, and D. Klingman, "Real-World Applications of Network Related Problems and Breakthroughs in Solving them Efficiently," ACM Transactions on Mathematical Software, 1, 1, 1975.

14. F. Glover, and J.M. Mulvey, "Equivalence of the 0-1 Integer Program to Discrete Generalized and Pure Networks," to appear in Operations Research, 1977.

15. B. Golden, "Shortest Path Algorithms: A Comparison," Research Report OR 044-75, Massachusetts Institute of Technology, 1975.

16. D. Karney and D. Klingman, "Implementation and Computational Study on an In-Core Out-of-Core Primal Network Code," Operations Research, 24, (1976).

17. J.M. Mulvey, "Coordinating Database Design and Network Optimization," presented at the XXIII International TIMS meeting, Athens, Greece, July, 1977.

18. E. Moore, "The Shortest Path Through a Maze," Proceedings of the International Symposium on the Theory of Switching, 1957.

19. U. Pape, "Implementation and Efficiency of Moore-Algorithm for the Shortest Route Problem," Mathematical Programming, 7 (1974), 212-222.

20. D.W. Robinson, "Analysis of a Shortest Path Algorithm for Transportation Applications," Control Analysis Corporation, Technical Report, March, 1976.

21. E.K. Yasaki, "The Mini: A Growing Alternative," Datamation, May, 1976, 139-142.

22. T. Ziever, W. Mitchell, and T.R. White, "Practical Applications of Linear Programming to Shell's Distribution Problems," Interfaces, 6, 4, 1976.

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)
Center for Cybernetic Studies
The University of Texas

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

3. REPORT TITLE
(6) An Evaluation of Mathematical Programming and Minicomputers.

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

5. AUTHOR(S) (First name, middle initial, last name)
(10) Joyce/Elam
Darwin/Klingman
John/Mulvey

(9) Research rept.

6. REPORT DATE
(11) Oct[redacted]77

(12) 24 p.

7a. TOTAL NO. OF PAGES
19

7b. NO. OF REFS
22

8a. CONTRACT OR GRANT NO.
N00014-76-C-0383 and
N00014-75-C-0569;0616 ; DOT-OS-70074

b. PROJECT NO.
NR047-021

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)
Center for Cybernetic Studies
Research Report CCS 313

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
(14) CCS-313

10. DISTRIBUTION STATEMENT
This document has been approved for public release and sale;
its distribution is unlimited.

(15) N00014-76-C-0616, N00014-75-C-0569

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY
Office of Naval Research (Code 434)
Washington, DC

13. ABSTRACT

The purpose of this paper is to explore the viability of implementing mathematical programming algorithms on minicomputers. We feel that the minicomputer explosion presents many exciting research challenges in terms of both identifying and matching appropriate algorithms and applications for such hardware. This paper partially examines these issues by comparing the computational performance of two shortest path algorithms on four very distinct types of computer hardware.

406 197

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6811

Unclassified

Security Classification

A-31408

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Shortest Path						
Traffic						
Networks						
Minicomputers						