THE USE OF COMPUTERS IN INDUSTRIAL ENGINEERING EDUCATION

Richard C. Wilson

Department of Industrial Engineering
The University of Michigan

January 27, 1962

This material is distributed by The Ford Foundation Project on the Use of Computers in Engineering Education at The University of Michigan. This report appears in the library edition of the Final Report of the Project and is also issued as a separate booklet. Similar "Curriculum Reports" for other engineering disciplines are also available on request.

ABSTRACT

During the past two years, the Industrial Engineering Department at The University of Michigan has been exploring the use of computers in its undergraduate courses. Six faculty members have taken part in the faculty training programs of the Project on the Use of Computers in Engineering Education. In addition, twelve Industrial Engineering professors from other universities have participated in the Project's activities for periods of from one week to a full semester.

All Industrial Engineering undergraduate students at the University are required to take an introductory sophomore level digital computer course taught by Computing Center and Mathematics Department personnel. Digital and analog computer work has been regularly assigned in five departmental courses to give students an opportunity to gain practice in the application of computers in the solutions of their engineering problems. One of these, a required course in data processing, has just recently been incorporated into the curriculum. The Industrial Engineering curriculum, the areas where computer activity has been most successful, and the current and future impact of computers on industrial engineering are discussed.

In addition to the general University computing facilities, the Department has its own laboratory with two LGP-30 digital computers which have been used extensively for production-inventory game-type simulations.

A selected set of eight example problems prepared by faculty Project participants are included. These may be considered as a supplement to the 56 example engineering problems which have been published previously by the Project.

Table of Contents

		Page
I.	Introduction	A3
II.	The Impact of Computers on Industrial Engineering Technology	A6
III.	The Impact of Computers on Education	A7
IV.	Examples of Application of Computers to an Industrial Engineering Curriculum	A13
V.	Experience with Computer Types and Sizes	A18
VI.	Summary	A20
VII.	Sample Problems Prepared by Faculty Participants	A21
	57 A Queueing Dynamics Problem	A22
	58 Optimizing Machine Loading	A31
	59 Regression Analysis	A37
	60 Critical Path Scheduling	A42
	61 Minimum Path Through a Network	A48
	62 Engineering Economy	A54
	63 Traffic Count Analysis	A59
	64 Machine Utilization	A67
	Bibliography	A69

TABLES

IA	Industrial Engineering Department Staff	A4
IIA	Computer Content of 1957-58 and 1961-62 Industrial Engineering Department Courses	A5
IIIA	How Computers Contribute to Industrial Engineering Education	A7
IVA	Heuristic Problem Solving Steps	A10
VA	Course Outline: Data Processing	A14

USE OF COMPUTERS IN INDUSTRIAL ENGINEERING EDUCATION*

I. INTRODUCTION

Recent developments in Industrial Engineering have been paralleled by the development of the computer as an important adjunct to the modern engineer. As we will show, the interplay between Industrial Engineering practice and computer technology already has been considerable, but the full impact of computers on Industrial Engineering education is still to come. The Ford Foundation Project on the Use of Computers in Engineering Education has enabled the Industrial Engineering Department at The University of Michigan to experiment in depth with the use of digital computers in its curriculum. Although a number of impressions can be reported, it is important to emphasize that these are based on work-in-progress rather than completed effort. Ideas for extension of reported progress are many and depend mainly upon the available time and creativity of the Industrial Engineering profession. Only after an Industrial Engineering student has completed his full undergraduate schooling in a computer oriented atmosphere, can the impact of computer innovations be fully evaluated.

During the last several years the Industrial Engineering Department at The University of Michigan has enjoyed tangible support and encouragement of its exploration of computer utilization in depth. Many different factors and groups have contributed to this environment. The Computing Center of the University has furnished the highest level of technical help to all students and faculty who use its IBM 709. The Center's open-shop policy has encouraged use of its machine by all University units. In addition, the development of the Michigan Algorithmic Decoder (MAD) Compiler and the many special cours offered in compiler use continue to permit neophytes to approach a computer with a minimum of effort. The Ford Foundation Project has contributed in many ways to the computer-oriented environment of the Engineering College which has led the faculty to accept the computer as a tool. The experimental objectives of the Project encouraged faculty members to attempt computer applications which they might otherwise have vigorously opposed. By furnishing technical personnel, manuals, and counsel when requested, the Project assisted the Industrial Engineering Department in initial use of its own LGP-30 computer. Six University of Michigan and five visiting Industrial Engineering faculty members have been supported by Project funds in order to study computers. Experience in educating faculty in the use of computers is reported in the Second

---

* This report is possible only because of the valuable discussions with many different persons throughout the duration of the project. Appreciation for their comments (not necessarily in agreement) are due W. Allen, E. P. Dandridge, J. T. Elrod, W. Hancock, D. L. Katz, and D. H. Wilson.

Annual Report of the Ford Foundation Project.[12]

This report is, therefore, a composite of views which have arisen in an academic environment conducive to the application of computers whenever conceivable. If this environment is too strongly computer-oriented we are too close to recognize it; we will show, however, that one striking benefit has been a concurrent increase in the analytic content of undergraduate and graduate courses. This increase continues without sign of abatement. Although this report uses The University of Michigan's Industrial Engineering curriculum for its specific examples, we believe other Industrial Engineering Departments will also find our experiences valuable in curricula planning.

A brief description of the Industrial Engineering Department at The University of Michigan will give the reader a frame of reference for the remainder of the report. During the 1961-62 academic year approximately 240 undergraduates are working for a B.S. degree in the four-year Industrial Engineering program. An additional thirty students are working toward Master's degrees and twenty others are working toward the Ph.D. in Industrial Engineering. The departmental staff (see Table IA) consists of the Chairman, eight full time faculty, four part-time faculty and three Teaching Fellows. Eight of the staff currently use the computer in connection with their teaching or research. The course requirements for the B.S. in Industrial Engineering are given in Table IIA for both the 1957-58 and the 1961-62 programs. A comparison of these two undergraduate programs reveals that the number of courses requiring the use of computers has increased in the five years from none to six, not counting electives or occasions when the student voluntarily turns to the computer. The graduate program has experienced an equally dramatic increase.

TABLE IA

Industrial Engineering Department Staff

Wyeth Allen, B.M.E., D.Eng., Professor of Industrial Engineering and Chairman of the Department
Merrill M. Flood, Ph.D., Professor of Industrial Engineering, and Senior Research Mathematician, Mental Health Research Institute
Robert M. Thrall, Ph.D., Sc.D., Professor of Operations Analysis, Department of Industrial Engineering and Professor of Mathematics, College of Literature, Science and the Arts
James A. Gage, M. S., Professor of Industrial Engineering
Quentin C. Vines, M.E., Associate Professor of Industrial Engineering
Wilbert Steffy, B.S.E., Associate Professor of Industrial Engineering
Edward L. Page, M.S.E., Associate Professor of Industrial Engineering
Clyde W. Johnson, A.B., Associate Professor of Industrial Engineering
Walton M. Hancock, D.Eng., Associate Professor of Industrial Engineering
Richard W. Berkeley, B.S.E., Assistant Professor of Industrial Engineering
Richard V. Evans, D.Eng., Assistant Professor of Industrial Engineering and Assistant Professor of Mathematics, College of Literature, Science and the Arts
Richard C. Wilson, Ph.D., Assistant Professor of Industrial Engineering
Dean H. Wilson, B.S., Lecturer in Industrial Engineering and Research Engineer, Institute of Science and Technology

---

[12] Superscript numbers refer to references, page A69.

TABLE IIA

Comparison of Computer Content of 1957-58 and 1961-62 Programs
of The University of Michigan Industrial Engineering Department

Course Requirements
(Exclusive of Engineering College Common Core)

1957-58		1961-62	
Course Title	Hours	Course Title	Hours
English	4	English	4
Drawing	2		
Economics	3	Economics	3
Engineering Mechanics	10	Engineering Mechanics	7
Thermodynamics	5	Thermodynamics	3
Machine Design	6	+ Machine Design	7
Materials and Processing	9	Materials and Processing	9
Industrial Management	3	* Industrial Management	3
Personnel and Wages	5	Personnel and Wages	5
Plant Layout, Materials Handling	3	Plant Layout, Materials Handling	3
Methods and Work Measurement	3	Methods and Work Measurement	3
Production and Inventory Control	2	* Production and Inventory Control	2
Operations Research	3	* Operations Research	3
Engineering Economy & Accounting	8	Engineering Economy & Accounting	8
Statistics	6	Statistics	6
Electrical Engineering	4	Electrical Engineering	4
		* Computers & Data Processing	4

* Requires use of digital computer.
+ Requires use of analog computer.

In this report we first outline ways in which computer capabilities have introduced new technology into Industrial Engineering and then point out how these same capabilities can change the Industrial Engineering educational process itself. Our main theme is that computers suggest a structure for thinking and a language of communication. The specific impact of this theme is then illustrated by examples of the current use of computers in The University of Michigan Industrial Engineering curriculum, including problems, specific computer programs, and evaluation of the educational result. Next follow brief observations regarding the educational merits of departmental vs. large central computer facilities. A role for the analog computer in Industrial Engineering is also suggested. The report concludes with observations regarding the ideal role of computers in Industrial Engineering education.

II. THE IMPACT OF COMPUTERS ON INDUSTRIAL ENGINEERING TECHNOLOGY

The rapidly enlarging capabilities of computers have already significantly enlarged the scope of practical and theoretical problems which confront the Industrial Engineer. These capabilities constitute the computer technology with which graduating Industrial Engineers must be familiar. Curriculum planners must therefore give serious consideration to what, how much, and where instruction in computer technology is to fit into the Industrial Engineering curriculum. Effective consideration requires understanding the broad capabilities and current limitations of digital computers. A general review of computer technology can be found in the I.R.E. Computer Issue.[1] Armed with this understanding, the curriculum planner can recognize changes in the Industrial Engineering curriculum which computer capabilities make desirable. For example, four broad capabilities of digital computers are given in Table IIIA. Each of these four capabilities has immediate relevance to Industrial Engineering by extending the scope for practical application of many heretofore "esoteric" concepts. It is well known that only the use of computers makes practical the computation of linear programming problems with perhaps hundreds of variables. The possibility of computation-induced round-off errors in these problems introduces the need for Numerical Analysis concepts as a part of the practical application of computers. Some of the technology, such as Numerical Analysis, which has become relevant to Industrial Engineering because of computer capabilities, is shown in Column II of Table IIIA.

Much of this material already appears in Industrial Engineering Curricula, but the rapid change of technology will not permit educators the luxury of a status quo. For example, Leubbert[2] suggests the importance of integrating data-processing concepts with information-processing equipment design and operation, or with the development of assemblies for communications systems, surveillance systems, and process control systems. He argues that the Electrical Engineering departments need to broaden their perspective in the direction of data processing. By the same argument, Industrial Engineering needs to expand its scope toward information system technology. Are Industrial Engineering curricula sufficiently concerned, for example, with the importance of engineering developments in information processing equipment on the design of warehousing information and data-processing systems, or on the design of manufacturing control information systems? What are the implications of developments in machine learning described by Gravell[3] and Minsky[4] on the manufacturing control systems of the future? These and similar questions for Industrial Engineering curriculum growth suggest important further impacts which computers will have on Industrial Engineering technology.

Use of Computers in Industrial Engineering Education

TABLE IIIA

How Computers Contribute to Industrial Engineering Education

I	II	III	IV
Digital Computer Capability	New Technology[1] Pertinent to I.E.	Educational Methodology Suggested by Capability	Principle of Learning Utilized
Performs Simple Repetitive Computations Rapidly	Information Retrieval, Data Processing and Information Systems, Sorting	1. Flow Diagramming 2. Math. Notation	Organization of Material, Activity
Expands Size of Complex Numerical Problems which are Feasible for Solution	Multi-Variate Statistics, Linear Programming, Numerical Analysis	3. Use of Real Size Problems 4. Numerical Solution of Complex Abstractions	Feedback Activity Variety
Permits Iterative Approximation of Intractable Numerical Problems	Scheduling Theory, Search Theory	5. New Approach to Previously Ignored Problems 6. Explore Solutions Parametrically	Variety Feedback Activity
Permits Study of Complex Logical (Non-Numeric) Problems	Machine Tool Control, Pattern Recognition, Simulation, Logic Theory	7. Visual Dynamic Displays 8. Logical Model Development	Organization Feedback Contiguity
	Process Control Servo Theory Game Theory Learning Theory	9. Laboratory Simulation Man-machine Experiments 10. Interdisciplinary Focus	Activity Variety

III IMPACT OF COMPUTERS ON EDUCATION

We accept the premise that computer technology is essential content for Industrial Engineering curricula; we now turn to the related question: What impact does the computer have on the educational process itself? The effectiveness of engineering teaching is in part related to the understanding and utilization of principles of learning by the teaching staff. McKeachie[5] proposes these principles for efficient learning:

a. Feedback: Knowledge of achievement and expected progress must be available to the student.

b. Contiguity: Feedback or appraisal should be as immediate as possible to the test or trial.

c. Activity: Participation, action, or student response improves learning.

d. Organization: Presentation of related ideas in logical sequence simplifies learning.

e. Variety: A wide variety of practice problems yields better ability to apply learning to new situations.

Column III of Table IIIA lists some educational methodology suggested by the digital computer capability in Column I. In order to illustrate the usefulness of each methodology to the learning process, Column IV lists some of McKeachie's principles of learning which are utilized in each of the methodologies. Let us examine more fully the relationship between the educational methodology, principles of learning, and computers.

Flow diagramming, a schematic representation of relationships between activities, is used in some form by many industrial engineers. Its usefulness in planning and communicating a computer program for a data-processing activity or mathematical algorithm reconfirms the importance of the device for organization of material. Computer programming, therefore, offers useful experience to the student in organizing his attack on problems. Universally, experienced faculty feel that one of the important benefits of programming a computer is the logical, complete, and unambiguous algorithm for problem solution which must be constructed. To write a working computer program, students must prepare a more explicit and general statement of the solution method than is required to calculate the answer by hand. As a result, both faculty and student are forced to communicate by precise mathematical notation where heretofore loose description was often sufficient. Payroll procedures, inventory control systems, balance sheet preparation, standard data development must all be stated with mathematical precision in order to be programmed. (The Ford Foundation Project's First Annual Report[15] presents programmed examples.)

We believe the contributions of flow diagramming and mathematical notation to the learning process to be of major significance. A scanning of articles and speeches on the present "crises" in engineering education reveals a general agreement that (a) the curriculum leading to an engineering Bachelor's degree does not permit sufficient time to cover all the significant facts related to a particular discipline; (b) a return to the fundamentals of engineering education is mandatory, with simultaneous weeding out of courses devoted to description of contemporary practice; (c) one way to return to fundamentals is to introduce more science into the engineering course content. Similar agreement about the science content fundamental to engineering education is notably lacking. Teaching the "engineering way of thinking" is a universally accepted objective. One inference to be drawn is that the subject matter content of the curricula may be of secondary importance to the teaching of the "engineering way of thinking." Arguments that more basic science content improves this indoctrination in engineering thinking are not well supported.

Instead, investigations reported by Hatch and Bennet[6] suggest that learning is improved by "problem orientation" rather than by subject matter orientation. They point out that "when teaching and learning are made forms of inquiry . . . both the teacher and the student apparently still need help with the specifics . . . How does one ask questions?

What is an effective pattern of progression in questioning? The object of research on effectiveness of teaching should shift from the tactics of teaching to the logistics of learning, to methods which, in contradiction to the pedagogical, may be described as the methods of scholarship, of inquiry, of problem solving, or of critical thinking." In addition they say[7]: "Students who seek their own answers, sometimes to their own questions, not only learn more, but become better learners. Furthermore, problem solving stimulates the life for which the student is preparing himself. Since the engineer in today's classroom cannot learn enough to meet all the demands that even his first job will make upon him, and since he cannot be instructed in what he should do with machines that have not yet been designed, he needs to learn and to learn early how one addresses oneself to problems, particularly unanticipated problems. . . . This is presumably the reason for the new emphasis on principles and on theory in engineering . . ." Others support this point of view. Balabanian[8] says: "One of the most important objectives of any technical engineering course should be the exposition of sound engineering thinking. . .the scientific method." Both Balabanian and Russell[9] argue that all engineers should be familiar with logical thinking, including formal training in terminology and logical relations.

In Table IVA, we illustrate the surprising analogy between the four major aspects of heuristic problem solving suggested by Polya[10] and the elements of activity involved in programming and solving a problem by use of a digital computer. The Critical Path Program (Problem 60,* page A22) is used for this illustration. Briefly stated, the problem assumes a project composed of a number of tasks, some of which must be completed before others are begun. Associated with each task is a time for performance. The objective is to find the minimum time for completion of the project and to discover those tasks which are critical to meeting the minimum time. The problem can be solved very simply if the number of tasks to be scheduled is few. A quickly devised plan of solution might be to list all possible sequences of tasks from start to completion; the sequence with the longest total completion time is the "critical path." Such a plan, however, is impractical for problems with a large number of tasks or for calculating critical paths for many different projects. Further, it is neither sufficiently explicit for someone else to understand, nor sufficiently thought out to be surely general.

---

* Problem numbers refer to the example problems prepared by faculty participants in The Ford Foundation Project and are found in Section VII, page A21.

TABLE IVA

HEURISTIC PROBLEM SOLVING STEPS

I. Understanding the Problem

What are the Data?

What are the conditions on the data?

What are the Unknowns?

Earliest start, Earliest finish, Latest start,
Latest finish, Total float for each task

Introduce Suitable Notations

Knowns: Number of tasks = N
Initial node of task K = I(K)

Unknowns: Earliest Start = ES(K) = TI(I(K))
Earliest Finish = EF(K)
Latest Start = LS(K)
Latest Finish = LF(K) = TE(J(K))
Total Float = TF(K)

What are the conditions linking knowns and unknowns?

Draw a figure.

II. Devise a Plan

A. Begin at initial node; find the maximum of the earliest finish times at each connected node; continue through all nodes.

B. Begin at final node; find the minimum of the latest start times at each connected node; continue through all nodes back to initial node.

C. Compute all unknowns from conditions.

III. Carry Out the Plan

A. Check each step.

B. Can you prove it is correct?

Run a check program until debugged, and correct results are obtained.

IV. Examine the Solution

A. Can you derive the results differently?
Can a better program be written?

B. Does the method work for some other problem?
What are the restrictions on use of the program?
Does the program work on other data?

COMPUTER PROGRAM

```
$DATA
      21
   1   2   LEAD TIME              10
   1   5   LINE AVAILABLE         28
   2   3   MEASURE, SKETCH         2
   3   4   DEV. MAT'L LIST         1
   4   5   DUMMY                   0
   4   6   ERECT SCAFFOLD          2
   4   7   PROCURE PIPE           30
   4   8   PROCURE VALVES         45
   5   6   DEACTIVATE LINE         1
   6   8   DUMMY                   0
```

Data, Continued

```
   6   9   REMOVE OLD PIPE         6
   7   9   PREFAB SECTIONS         5
   8  11   PLACE VALVES            1
   9  10   PLACE NEW PIPE          6
  10  11   WELD PIPE               2
  11  12   FIT UP                  1
  11  13   INSULATE                4
  12  13   DUMMY                   0
  12  14   PRESSURE TEST           1
  13  14   REMOVE SCAFFOLD         1
  14  15   CLEAN UP                1
```

```
$ COMPILE MAD, EXECUTE, DUMP, PRINT OBJECT, PUNCH OBJECT          CRCL
      EXTERNAL FUNCTION (N,I,J,DIJ,ES,LS,EF,LF,TF,FF,TI,TE)
      ENTRY TO CRTCUL.
      INDEX=1
      THROUGH C, FOR K=1, 1, K .G. N
      WHENEVER I(K) .GE. J(K), TRANSFER TO OUT 1
      WHENEVER I(K) .L. INDEX, TRANSFER TO OUT 2
      WHENEVER I(K) .G. INDEX .AND. I(K) .NE. INDEX +1, TRANSFER TO OUT 3
C     WHENEVER I(K) .E. INDEX +1, INDEX = I(K)
      EXECUTE ZERO.(TI,TE)
      THROUGH A, FOR K = 1, 1, K .G. N
      MAX = TI(I(K)) + DIJ(K)
A     WHENEVER TI(J(K)) .E. 0 .OR. TI(J(K)) .L. MAX, TI(J(K)) = MAX
      TE(J(N)) = TI(J(N))
      THROUGH B, FOR K=N,-1, K .L. 1
      MIN=TE(J(K))-DIJ(K)
B     WHENEVER TE(I(K)) .E. 0 .OR. TE(I(K)) .G. MIN, TE(I(K))=MIN
      THROUGH D, FOR K=1, 1, K .G. N
      ES(K)=TI(I(K))
      LS(K)=TE(J(K))-DIJ(K)
      EF(K)=TI(I(K))+DIJ(K)
      LF(K)=TE(J(K))
      TF(K)=TE(J(K))-TI(I(K))-DIJ(K)
D     FF(K)=TI(J(K))-TI(I(K))-DIJ(K)
      TRANSFER TO OUT
      NORMAL MODE IS INTEGER
OUT 1 PRINT FORMAT ERR1, K, K
OUT 2 PRINT FORMAT ERR2, K
OUT 3 PRINT FORMAT ERR3, K
      VECTOR VALUES ERR1=$1H1, 3H I(I3,18H) NOT LESS THAN J(I3, 1H)*$
      VECTOR VALUES ERR2=$1H1, 3H I(I3,17H) OUT OF SEQUENCE *$
      VECTOR VALUES ERR3=$1H1, 3H I(I3,9H) MISSING *$
OUT   FUNCTION RETURN
      END OF FUNCTION
```

In contrast, programming a solution algorithm for a computer does not permit the similar luxury of omitting essential steps to problem solving. The preparation of a computer program disciplines the student to rigorous adherence to all aspects of the problem-solving sequence. Clear definition of known and unknown data must be made at the start of the problem; suitable notation must be provided; the plan for solution must be explicit and unambiguous; oversights cannot be allowed; special conditions must be anticipated. The computer is a rigorous taskmaster; only perfection is satisfactory. Debugging a program is therefore the ultimate in checking each step to establish the correctness of the intended algorithm. After debugging, it becomes possible to easily rerun the program using other data and thereby to explore the generality of the algorithm. The programming process is seen not only to reinforce the heuristics of problem solving, but also to make use of McKeachie's principles for efficient learning:

a. Feedback: The program either runs successfully or fails due to careless grammatical or punctuation errors in the program, or because of faulty logic in the solution plan. In either case the fact that an error has been made is known only to the student.

b. Contiguity: The turn-around time for processing through the computer is short, usually under one day; the student can submit his program at any time and knows when it is correct. He need not wait until everyone in the class has turned in his work to have his own evaluated.

c. Activity: The student is fully involved at all stages of the problem, including finding and correcting errors.

d. Organization: Only correctly organized programs will run. The flow charting process assists in organization and completeness.

e. Variety: Successful programs can be used on a number of problems without additional programming effort. The student is stimulated to look for new and perhaps unique applications of programs because of the low cost (to him) of the additional experience. He also becomes aware of the many alternative ways of solving the same problem and often looks for ways to improve an existing program.

The digital computer, however, contributes more to learning than in just developing skills for algorithm construction. The availability of computers makes it feasible for students to solve real problems within the limitations of educational time. Carefully chosen examples with neat numerical results are no longer necessary. Instead, large-scale problems with unselected data can be given the student. Techniques for minimizing error generation in linear programming algorithms become important new knowledge to be learned. The design of multi-variate experiments is useful subject matter. Feedback is fast and involves no frustrating computation errors. After a program has been developed, the variety of problems which can be explored, by multiple regression, for example, is increased because of the elimination of computation tedium. The programming activity itself brings home to the student the usefulness of abstract mathematical notation, because

it becomes a means for precisely communicating a real problem. A sequence of exploratory solutions to linear programming problems becomes feasible. Previously intractable problems such as line balancing can now be explored through programmed search techniques. The use of digital computers for presenting visual dynamic displays is expensive, but with proper development should be an effective means of achieving contiguity.

The capability of the digital computer to explore not only numerical problems but also complex logical problems is among the most exciting aspects of computer technology. Models of logical, rather than numeric, processes can often be developed and used in education. In fact, the current languages of computer programming offer new ways of describing processes which have heretofore defied precise definition. A simulation program is such an example. New computer programming languages specifically oriented to industrial problems may soon greatly enlarge our ability to quickly and precisely define large industrial systems, just as matrix algebra has enlarged the scope of array manipulation. Management games are examples of logical models used to add variety and activity to the learning process. Through the use of such games, the significance of sociological and psychological aspects of industrial systems becomes real. As a result, the importance of interdisciplinary problem-solving becomes apparent to the student. The computer thus becomes a focal point or common communication medium for many separate disciplines. Methods of problem solution used in one field or discipline may be found useful in others. For example, the labeling process used in the algorithm for the network best-path problem (Problem 60, page A42) is also utilized in a truss analysis program for determining stress in trusses by the method of joints (see the Civil Engineering portion of this final report, Example Problem: "A MAD Program for Truss Analysis" by K. H. Chu). Consolidation of effort rather than duplication of study on similar problems also occurs.

Our experience has been that students willingly devote large amounts of their time in preparation and programming problems of their own choosing for the computer. We believe that the students discover that the process is imbued with many of the principles of learning and hence instinctively find it satisfying, even if frequently frustrating. If this is true, computers serve as useful adjuncts to the learning process by reinforcing and offering experience in the application of engineering thinking. The good teacher is constantly searching for additional means of instilling an understanding of the engineering method. Properly used, the computer is a powerful ally in this endeavor.

IV EXAMPLES OF APPLICATION OF COMPUTERS TO INDUSTRIAL ENGINEERING CURRICULUM

In order to conform to some degree of generality, we assume that with varying emphasis, Industrial Engineering Curricula contain subject material classified according to the A.I.I.E. Research Divisions:

Applied Mathematics	Facilities Planning	System Design and Synthesis
Applied Psychology	Data and Information Processing	Organization
Work Measurement	Production and Inventory Control	Engineering Economy
Materials Processing	Methods	

By using this classification in describing the potential contribution of computers to the Industrial Engineering programs, we minimize the description of the content of specific courses elsewhere.

Generalizations about the proper academic level for introduction of specific computer problems is meaningful only if details about the sequence of topics throughout a complete and specific program is presented. Reference therefore to academic levels is avoided insofar as possible, and subject areas are emphasized.

Experience with computers at The University of Michigan supports the introduction of computer technology and programming into the curriculum at the earliest opportunity. Currently, most University of Michigan engineering students are required to take a one-hour introductory programming course offered by the Mathematics Department in the second semester of the sophomore year (a text containing lecture notes for this course is available as of March, 1962[13]). The student's mathematical maturity is evidenced by the fact that concurrently he takes a four-hour mathematics course, one half of which is devoted to differential equations, and the other half to probability and statistics. Our experience indicates that this initial exposure to computers needs immediate reinforcement if the student is to develop sufficient facility to look upon the computer as a natural tool for problem solving before the end of his four-year program. Consequently, during the first semester of his junior year a data processing course is required by the Industrial Engineering Department for every student. The outline for this three credit course is shown as Table VA. At the end of this course, the student is expected to understand the logical structure of machines and the concepts for application of special or general purpose computers to many of the new technologies pertinent to Industrial Engineering (listed in TableIIIA, page A7). He is also expected to have sufficient skill to write his own programs in a compiler language for at least one of the machines which are available to him on campus. It is believed that this early exposure to computers affords the student a framework for building his own perspective and for gaining useful experience in determining those areas in which computers may have immediate and economic relevance. Discussion in later courses of industrial use of com-

puters is facilitated. No further time is taken from subsequent courses however for explicit discussion of programming techniques. Further, the omission of explicit programming discussion in subsequent courses permits the student to use his own judgment in deciding if a particular problem can be more effectively solved by the use of computers.

TABLE VA

Course Outline: Data Processing

Credit Hours: Three hours per semester, fifteen weeks

Contact Hours: Two one-hour lectures per week; one two-hour recitation per week

Text: Gregory and Van Horn: Automatic Data Processing Systems, Wadsworth Publishing Co., 1960.

Supplemental: IBM Applications Manuals. Richards, Arithmetic and Digital Operations, Van Nostrand, 1955, and M.A.D. Compiler Manual, The University of Michigan Computing Center

Topic	Number of Sessions
Processing Data by Machines	3
Number Systems, Circuit Logic, Programming Logic	3
Machine Operations and Arithmetic	3
Debugging and Checking Procedures	2
Flow Charting of a Payroll Application and Programming Example for IBM 709	3
Input, Output Equipment	3
Processing Procedures, Sorting, Tape Merging	6
Case Example: Production-Inventory Control System	3
Simulation Concepts:	
Random Number Generation, Sampling	12
Programming of Simulations	
Experimentation and Validation	
Example of Simulation Application	2
Optimization Concepts and Search Techniques	5

In Applied Mathematics, the computer has been invaluable to Industrial Engineering. A library program for multiple regression analysis is an Applied Statistics program which has many applications. The student able to prepare and use such a program adds considerable scope to the amount of analysis and size of experiments which he can undertake. An excellent example of Queueing Theory is found in the Queueing Dynamics Problem (Problem 57, page A22). Exploration of this problem is demonstrated by both the analog and digital computer. Finding the minimum-path-through-a-network is a problem particularly suited for computer programming by undergraduates because of the logical simplicity but highly iterative character of the procedure (Problem 61, page A48). Many other applied mathematics possibilities easily suggest themselves. The examples given, however, illustrate adequately

the degrees of difficulty associated with such algorithms. By careful selection, these and other programming problems can be effectively utilized to reinforce and evaluate student command of underlying theory.

The area of Facility Planning is especially amenable to effective use of computerized problems. The typical facility planning project is usually based on masses of data which, because of the difficulty of exhaustive analysis, are used for subjective evaluation of alternative proposals. Layout problems assigned to students must usually be abridged so they can be completed in the available time. Simplification often destroys the basic challenge of facility planning problems: the necessity to determine important factors of the problem from masses of information. Artificial or arbitrarily imposed simplification also restricts the depth of useful and instructive analysis. If students are competent to turn to the computer when they believe it useful, new possibilities arise for the development of both principles and practice in facility planning. The example problem on "Machine Utilization" (Problem 64, page A67) illustrates the ability of computer analysis to derive information too time consuming for hand computation. Thus, the number of machines required in a production facility and the appropriate choice of arrangement can be determined and evaluated in a manner never before feasible. Traffic Count Analysis (Problem 63, page A59) also illustrates how the digital computer can be brought to bear on analysis of data. Without a computer, the practical evaluation of the dynamics of operational systems as complex as most production facilities has been generally impractical. A single-station Monte Carlo simulation is easily within the competence of undergraduates, and more complicated multi-stage, multi-channel systems can also be run successfully if students have acquired reasonable competence in a general compiler language.

The character of work in Engineering Economy is particularly susceptible to major reorientation if a computer is readily available. Emphasis shifts rapidly to the mathematical structure of the available project evaluation and replacement models. Extensive work on interest computations and the use of interest tables is no longer justifiable since the computer makes parametric comparison of multiple alternatives not only possible but desirable. Once the principles are understood, tedious rate-of-return arithmetic will be avoided by computer-oriented students. The Rate of Return program (Problem 62, page A54) illustrates the ease of computer solutions. If the student is able to write a general program for a problem, his understanding is demonstrated. The implications of multiple roots in rate-of-return solutions must now be understood and their possible existence anticipated if the program is to be general. The development and application of broader

Engineering Economy models can progress if the present requirements for easy computation are relaxed. In fact, cost can often be included in computer programs for engineering design as the criterion of selection. Several examples of such programs are included in the Chemical Engineering portion of this final report.

Where alternative methods can be compared quantitatively, as, for example, by the use of MTM analysis, the computer can be applied instructively in Methods and Work Analysis by the student. Balancing of operations can be studied by simulation where analytical results may be unobtainable. Line balancing can also be carried out by computer processes for direct comparison with hand balances. A crew-size simulation problem, programmed by three graduate students for the LGP-30 computer, was a useful demonstration problem for discussion of Methods Study.

Several useful applications of computers in the area of Work Measurement are suggested by the example problems. Wider use of statistical tests, such as the "t" test used in validating and developing predetermined time systems is practical (see Reference 12, page 236). A simple linear regression is easily applied to a standards calculation (Problem 59, page A37). Multi-variate regression is introduced by contrasting the graphical development of multiple-factor time standards with a computer multiple regression of the same data. The perceptive student is quick to see two directions for work measurement research from a study of these applications: first, the possibility of using a computer to develop standards for new jobs, based upon stored data for each production department, and including, for example, variance estimates; second, the possibility of a real-time computer installation for collection of data and development of work standards. Recognition of the potential of such a system for control and scheduling and of the interrelationship among these activities follows logically.

The application of computers to Production Planning and Inventory Control teaching promises to be especially interesting. The program "Optimizing Machine Loading" (Problem 58, page A31) demonstrates a practical procedure for maximizing the loading of a critical machine. Critical path scheduling (Problem 60, page A42) and network flow (Problem 61, page A48) show how the astute choice of problems can result in continuity in the development of a student's programming skill and also in his deeper understanding of complex mathematical formulations of problem solutions. The student who attempts to translate a mathematical paper into an applied program will quickly learn to appreciate the value of developing facility with the literature of mathematics.

Further potential for raising the teaching level in this area of Industrial Engineering is furnished by computer technology. The availability of a compiler language which permits the preparation of closed subroutines or external functions independently of the main program

in which they are to be used offers great range for experimentation by the student. As part of this area, we have under development a simulation of a multi-product production-inventory system which uses externally prepared subroutines for the following decisions: dispatching priority, lot quantity, protective stocks, in-process inventory, finished goods stocks, purchase quantities, and order point. The student is given information about processing times, purchase lead times, demand estimates, and production sequence. He is asked to write and check out his own subroutines establishing the inventory policies. When these have been checked out and approved by the instructor, they are run as a part of the simulation in order that both the student and the instructor can observe the behavior of the system under the impact of the policies selected. In essence, the computer affords a method for evaluation of production-inventory policies under dynamic conditions.

Another interesting development is the use of a production-inventory simulation as a real-time laboratory for instruction in advanced production control. In this instance, the simulation is run on the departmental LGP-30 computer available for student operation. Information about the current status of the hypothetical operation is available to the student if requested, but only at an implied cost. Based on this information or on estimates on the part of the students, changes in policy can be implemented through the computer console, and experiments conducted during the operation of the simulation. Direct communication with the computer thus permits many of the dynamic aspects of real production-inventory flows to be brought into the classroom. In one sense, this might be catagorized as an inventory management game. In truth, however, the underlying structure of the simulation is so constructed that it may be altered to characterize a wide class of inventory-production systems. Further, the information sets are not rigidly established, but can be altered if instruction is thereby aided. The objective is clear: to study theoretical dynamic inventory-production models, to determine the appropriateness of such models for different systems, and to explore and develop new models where desirable.

In the teaching of Applied Psychology, Organization, and Materials Processing, application of computers has not thus far been made. No attempts have been made to "force" unnatural applications, and it is not anticipated that the computer will soon become an essential adjunct to education in these areas. In research in Applied Psychology, however, the availability of a computer does permit efficient analysis of experimental data. And in Materials Processing, recent developments in numerically-controlled machine tools motivate flow-chart analysis of the steps required for processing parts, thereby directly relating the logic of computer programming to the logical steps of machine tool operation. With these exceptions, the implications of computers for Materials Processing education or in Quality Control education have not been explored at Michigan. In the field of Organization

teaching also, unique contributions by use of computers have yet to be proven. However, interest in computers is considerably stimulated in the sophomore year when each Industrial Engineering student participates in a computerized Management Game. A competitive top-management game has been written for the LGP-30 computer. Up to eight teams may play; great flexibility is possible by an easy change of the economy tape or function constants. A detailed write-up of this game is forthcoming and will be distributed by the Ford Foundation Project. The game is utilized as an environment-rich laboratory in decision making and team organization, computationally feasible because of the computer's speed and accuracy. In this sense, the computer does contribute in a fresh way to the educational methods available to the instructor. Even though conclusive evidence that Organization Principles are learned from game playing has yet to be presented, we believe the experience is helpful. However, no evidence exists to indicate that complex computerized games are more effective teaching aids than simpler non-computerized games, or even role playing. In the area of Organization at the present time, therefore, we believe the computer may be a useful aid in research, but no conclusive evidence of its contribution to teaching in this area is known.

In all of the investigated areas of Industrial Engineering, the digital computer expands problem development in two distinct directions. On the one hand, it facilitates the generation of problem data according to some prescribed pattern, even possibly a random pattern. (See Optimizing Machine Loading, Problem 58, page A31) On the other hand, the ability of the computer to proliferate data in any given quantity strongly emphasizes to the student the needs and advantages of carefully planned experiments and statistical analysis of results.

V EXPERIENCE WITH COMPUTER TYPES AND SIZES

Departmental computer activity has made extensive use of both the Computing Center's IBM 709 and the department's LGP-30 computer. Introductory and advanced research programs have been developed by students and faculty for both machines. An evaluation of the role of small vs. large computers in the Industrial Engineering educational system is offered on the basis of this experience.

Student enrollment in the departmental course in Data Processing is currently between 30 and 50 students each semester. Use of the LGP-30 computer for special activities is seriously hampered by production runs imposed by large numbers of short, and frequently unsuccessful initial programming efforts. In spite of utilization of a high-speed reader to minimize input-output time and the availability of a second Flexowriter for off-line paper tape preparation, the LGP-30 has been found inadequate for permitting every student to prepare, debug, and run three problems of his own during the semester. The Computing Center's IBM 709 facilities, with more powerful compilers, fast executive system, and more

thorough compiler diagnostics, are now used for this phase of the education of the students. Those students who then have a particular need for direct communication with the computer during program runs easily make the transition to programming the smaller machines. The large machine capability is therefore highly desirable where the primary need is for introducing large numbers of students to computer programming. For the analysis of extensive research data, and for the solution of extremely large scale problems such as linear programming, the large computer is to be preferred.

In contrast, the departmental machine has decided advantages when considered as a special purpose facility and offers unique opportunities for Industrial Engineering programs. The slow speed of the smaller machine can be an advantage where operator intervention is an intended part of the system under study. A specific example of the use of this concept of man-machine communication has already been described in the application to the production-inventory game-type simulation. The utilization of a machine for the extended periods desirable for a laboratory experiment is not likely to be compatible with the Computing Center requirements for production runs. The small machine has the further unique capability of becoming an integral part of the laboratory by direct connection to the equipment under test. In this way, immediate data reduction is conceivable, and as a by-product, real-time control of the experiment itself can be arranged. The possibilities of a small computer to simulate a production facility suggest the concept of the Industrial Engineering laboratory of the future. Production data, generated by a small scale computer and displayed on special output panels similar to "Telecontrol," for example, will simulate the factory operations. The exact model used in the simulation may be unknown to the student production manager. Data output as desired by the student will be collected and analyzed, possibly in a second small-scale computer, as specified in a student-developed control system. Adjustments will be made on the simulated factory either automatically or by student intervention. In this concept lies the practical realization of a dynamic Industrial Engineering laboratory without the need for production capability as such. In 1957, Malcolm said[11], "Methods can and have been developed whereby the student can learn through interrogating or experimenting with a broad model of the design problem, or system under study. If this concept were rigorously pursued, the inspirational value coupled with the more comprehensive insights instilled could do much to provide the much needed speed-up in engineering education." These two unique potentialities of departmental computers: (1) their application to extended time operation with operator intervention, and (2) their adaptation to special uses for data collection, display, and control are of such significance that we believe small machines will find increasing use in engineering colleges, even, or perhaps, especially, in those with large computing facilities as well.

The potential of analog computers for Industrial Engineering education is almost untouched. One of the values of analog computers is their ability to produce results in graphical or visual form. The digital computer does not readily furnish output in patterns except with cathode ray tubes; even then rapid parametric exploration of the system is not possible by console operation. System design and analysis, in the sense of investigation of the stability of dynamic systems, has been alluded to in production and inventory control, and also in Facilities Planning simulations, for example. The possibilities of analog computer utilization for teaching demonstrations of the dynamic characteristics of models of industrial engineering systems are almost unexplored. The computer solution of a Queueing Dynamics problem (Problem 57, page A22) illustrates the possibilities of digital vs. analog computers in this area. In our search for dramatic laboratory aids and teaching-machines the relatively small cost of analog computers is a strong inducement for a serious study of this equipment, either as it now exists or as modified for easier programming. Besides being useful in queueing dynamics, analog computers can be valuable in exploration of inventory system response, economic systems stability, and renewal processes.

VI SUMMARY

The impact of computers on the Industrial Engineering program is great, both in expansion of technical content and in the development of learning. We have seen a growing trend toward more rigorous problem formulation and a broader system approach to problems by those students with computer facility. The creativity of students permitted to use machines on unstructured and original problem areas has contributed to broadening of course content. The use of flow charting and programming of problems at an early stage of engineering education reinforces problem-solving skills. We look to the near future when all seniors <u>should</u> be able and all graduate students <u>must</u> be able to communicate their problems to a machine when appropriate. The ideal situation has computing assistants in each department to aid the faculty in research and teaching by communicating problems to the computer without the drudgery and detail now required. Such laboratory assistance will also be needed to aid students in the preparation of their analysis of laboratory experiments, and in the supervision and maintenance of the computing equipment itself.

We believe that, in perspective, the glamor of using the computer has now abated. In its place is a growing realization that the power of the Industrial Engineer is in his ability to interpret industrial phenomena in analytic terms so that they may be transmitted to the computer for solution or understanding. This needed ability introduces irrevocable new demands and opportunities for Industrial Engineering education for which all faculty must be prepared.

VII SAMPLE PROBLEMS PREPARED BY FACULTY PARTICIPANTS IN THE PROJECT ON THE USE OF COMPUTERS IN ENGINEERING EDUCATION

Each participant prepared solutions for example problems of his own choice which he felt would be suitable for use in a classroom at the graduate or undergraduate level. A list of some example problems is given below. All digital computer problems are written in the MAD language which is fully described in "A Computer Primer for the MAD Language."[14] It should be remembered that most of these problems represent the participant's first efforts at computer solutions and may not necessarily be very sophisticated.

List of Problems

Number*	Title	Author	Page
57	A Queueing Dynamics Problem	R. A. Brown	A22
58	Optimizing Machine Loading	E. T. Kirkpatrick	A31
59	Regression Analysis	J. T. Elrod	A37
60	Critical Path Scheduling	R. C. Wilson	A42
61	Minimum Path Through a Network	R. C. Wilson	A48
62	Engineering Economy	J. Pistrang	A54
63	Traffic Count Analysis	J. Pistrang	A59
64	Machine Utilization	D. D. Deming	A67

* These problems may be considered as a supplement to problems 1 through 45 published in The First Annual Report and 46 through 56 published in The Second Annual Report and are numbered accordingly.

Example Problem No. 57

A QUEUEING DYNAMICS PROBLEM

by

Robert A. Brown

Industrial Engineering Department

The Ohio State University

Course: I. E. Applications of Computers Credit: 3 quarter hours (2 lecture, 2 laboratory)

Level: 4th or 5th Year

Statement of the Problem

This problem is taken from a naval logistics situation, and is a simple example of the use of computers in evaluating alternative operational systems.

During an amphibious invasion, landing craft move between deepwater ships, lying off-shore, and the beach. The landing craft characteristically move in groups or "waves". For any given amphibious operations, a certain amount of men and material must be moved within a critical time period or the operation can be presumed to have failed. Clearly, the total number of trips necessary to move the required tonnage can also be used to determine success or failure if the capacity of the landing craft is known. If, in addition, the time to make a round trip is known, then the number of landing craft required for the whole operation is determined provided that none break down or are lost to enemy action.

In reality the landing craft are subject to two kinds of hazard - a failure hazard and an attrition hazard. Attrition occurs because a craft is sunk or damaged beyond repair and is thus lost to the system. Failure occurs whenever the full operating capability of a craft is impaired so that it requires attention in a repair facility before being returned to service. To keep the problem simple we will treat both these hazards as constant percentages of the number of units in service at any time. (In reality they are likely to be time-dependent random variables.)

The repair facility will be limited in the number of landing craft it can process in unit time. Again, we will make the simplifying assumption that the repair facility will handle all demands for repair up to some maximum rate. For demands in excess of the maximum rate a queue or waiting line will form, and the output will be at the maximum rate.

Now both the landing craft and the service facility occupy shipping space. If there must be a minimum number of landing craft in operation during the critical period, how can this number best be ensured, by including a repair facility, or by carrying extra craft to replace all damaged and failed ones? Should we repair, or should we discard and replace? This question can be answered by providing information about the dynamic behavior of the system using both methods if costs and space requirements are known. We assume or estimate the attrition and failure hazards and the maximum repair rate, insert these values in a model of

the system, and compare the resulting costs and space requirements.

To summarize the operation of the system, the landing craft move in waves from the ships to the beach where they are subject to attrition. Of those returning to the ships, a portion, equal to the failure hazard, arrive at the repair facility for service while the remainder make up another wave to go into the beach. If the number to be serviced is less than the maximum repair rate, they are serviced and returned to augment the next wave; otherwise those in excess of the maximum repair rate form or add to an existing queue.

Set up the equations describing the operation of the system and solve them on the computer for the following values of the parameters:

Attrition hazard	A = 0.01, 0.05, 0.10
Failure hazard	F = 0.01, 0.05, 0.10, 0.20
Maximum repair rate	R_{max} = 5, 10, and 20% of the total initial force

It will be found helpful to normalize all variables at the outset. Don't forget that it is good practice to provide check solutions; these are usually based on letting the problem parameters take on extreme values for which the solutions can be found analytically.

Instructor's Solution

Two solutions, one analog and the other digital, are presented here. Beside giving an opportunity to learn the mechanics of programming either kind of computer, this problem has been designed to present an example of the method of approach to problems amenable to computer solution. In this connection, the importance of the state diagram which follows as a succinct description of the flow processes involved cannot be overemphasized.

The following symbols will be used:

T	Time
A	Attrition hazard
F	Failure hazard
RMAX	Maximum repair rate
N	Number of units in service
Q	Number of units awaiting repair (queue length)
R	Number of units returned to service from the repair facility (repair rate)

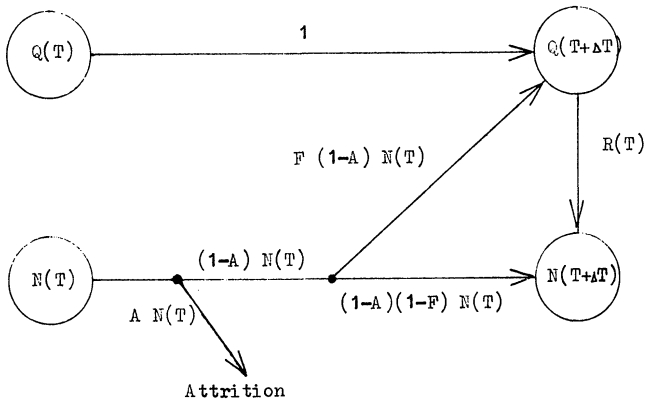
In addition, in the digital solution of the problem, two quantities are defined based upon extreme conditions to act as check solutions.

These are:

M	Number of units in service in the absence of failure (unlimited repair capability)
P	Number of units in service without repair or replacement

The time variable, T, is normalized by considering actual times relative to the time required to make one round trip to the beach. This is quite natural since the landing craft commonly move in "waves", thus discretizing the time variable. All quantity variables (M, N, P, and Q) and R are expressed as a percentage of the initial (T = 0) force, which again conveniently normalizes these variables. In the analog solution, the use of 100 volts to represent 100 percent yields an especially simple amplitude scale factor.

The quantity variables, corresponding to physical displacements, are represented on the state diagram by circles. The flow variables, corresponding to physical rates, are represented by arrows. The basic state diagram appears below. From this diagram the



State Diagram for Queueing Problem

following relations may be written; these are exact and, because they take the form of recurrence relations, may be used directly to set up the flow diagram for the digital computer. ($\Delta T = 1$ by the above normalization.)

$$N(T) = R(T) + (1-A)(1-F)\ N(T-1)$$

$$Q(T) = Q(T-1) + F\ (1-A)\ N(T-1) - R(T)$$

$$R(T) = \begin{cases} RMAX\ , & Q(T-1) > RMAX \\ Q(T-1), & Q(T-1) \leq RMAX \end{cases}$$

The quantities for check solutions are:

$$M(T) = (1-A)\, M(T-1)$$

$$P(T) = (1-A)(1-F)\, P(T-1)$$

For analog solution the following equations are written.

$$N(T+\Delta T) = N(T) - A\, N(T)\Delta T - F\,(1-A)\, N(T)\Delta T + R(T)\Delta T$$

$$Q(T+\Delta T) = Q(T) + F\,(1-A)\, N(T)\Delta T - R(T)\Delta T$$

$$R(T) = \begin{cases} RMAX, & Q(T) > RMAX \\ Q(T), & Q(T) \leq RMAX \end{cases}$$

Taking limits in the usual fashion yields:

$$\dot{N} = \Big((1-A)(1-F) - 1\Big) N + R = -AN - \dot{Q}$$

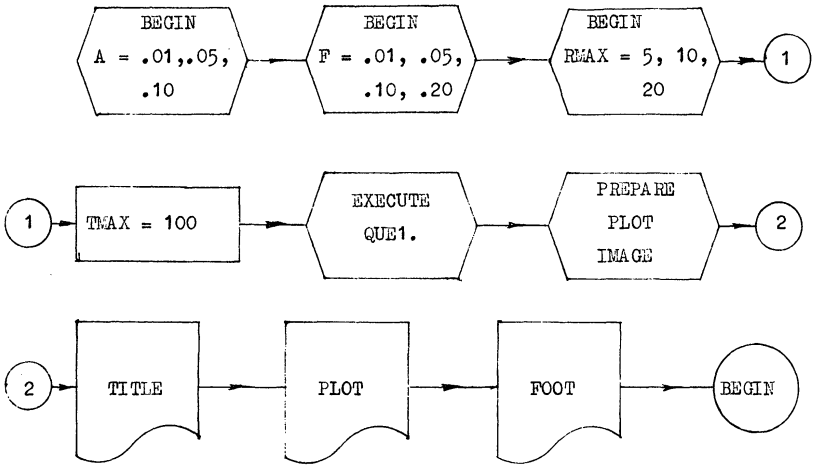
$$\dot{Q} = F\,(1-A)N - R = F\,(N-AN) - R$$

$$R = \begin{cases} RMAX, & Q > RMAX \\ Q, & Q \leq RMAX \end{cases}$$

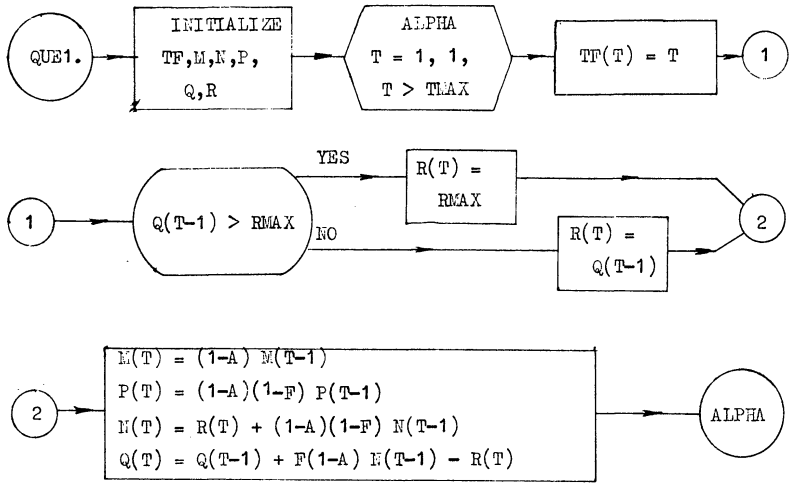
Flow Diagrams

A flow diagram for the digital computer and a circuit diagram for the analog computer appear below.

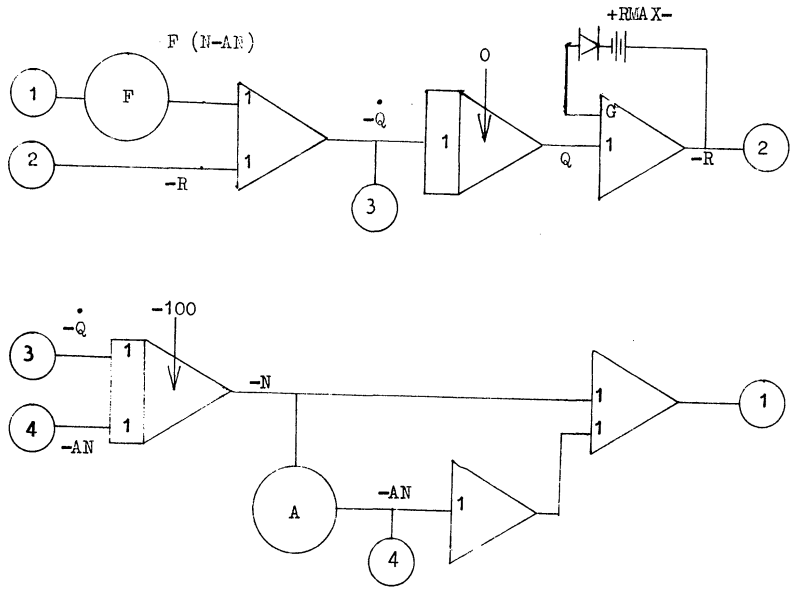
Digital Flow Diagram for Main Program.



Digital Flow Diagram for Subroutine QUE1.



In the digital computer solution, use was made of the plotting subroutine (PLOT.) available at The University of Michigan for the IBM 704. If this subroutine is unavailable, it is a simple matter to change the main program to print the numerical information, and subsequently to plot the results by hand. The actual computation of M, N, P, Q, and R is handled as an external function or subroutine to facilitate such changes. (TMAX is the maximum number of time intervals or "waves"; TF is the floating-point version of T. TF is required to be in floating-point mode for the use of the plotting subroutine mentioned above.)



Computer Diagram for the Analog

Example Problem No. 57

In the analog computer diagram several extra amplifiers have been used to obtain "one-knob" control of the problem parameters.

MAD Listing for Main Program

```
$COMPILE MAD, EXECUTE, PUNCH OBJECT, DUMP
R
R    QUEUEING PROBLEM
R     R. A. BROWN
R
 DIMENSION N(101),Q(101),R(101),TF(101),IMAGE(1000),M(101),P(1
101)
 INTEGER I,T,TMAX
 THROUGH BEGIN, FOR VALUES OF A = .01, .05,.10
 THROUGH BEGIN, FOR VALUES OF F = .01, .05, .10, .20
 THROUGH BEGIN, FOR VALUES OF RMAX = 5., 10., 20.
 TMAX = 100
 EXECUTE QUE1. (N,Q,R,M,P,A,F,RMAX,TMAX,TF)
 TFMAX = TMAX
 EXECUTE PLOT1. (NSCALE,5,10,5,20)
 VECTOR VALUES NSCALE = 1,0,-1,0,-1
 EXECUTE PLOT2.(IMAGE,TFMAX,0.,100.,0.)
 EXECUTE PLOT3. ($M$,TF,M,TFMAX)
 EXECUTE PLOT3.($P$,TF,P,TFMAX)
 EXECUTE PLOT3.($Q$,TF,Q,TFMAX)
 EXECUTE PLOT3.($R$,TF,R,TFMAX)
 EXECUTE PLOT3. ($N$,TF,N,TFMAX)
OUTPUT  PRINT FORMAT TITLE, A, F, RMAX
 VECTOR VALUES TITLE = $1H1,S10,63H      PLOT OF THE SOLUTION
1OF A QUEUEING DYNAMICS PROBLEM.  A =F6.4,5H, F =F6.4,
29H, AND R =F9.4*$
 EXECUTE PLOT 4.(12,LABEL)
 VECTOR VALUES LABEL = $    PER CENT$
BEGIN   PRINT FORMAT FOOT
 VECTOR VALUES FOOT = $1H0,S60,8HTIME---)*$
 END OF PROGRAM
```

Mad Listing for Subroutine QUE1.

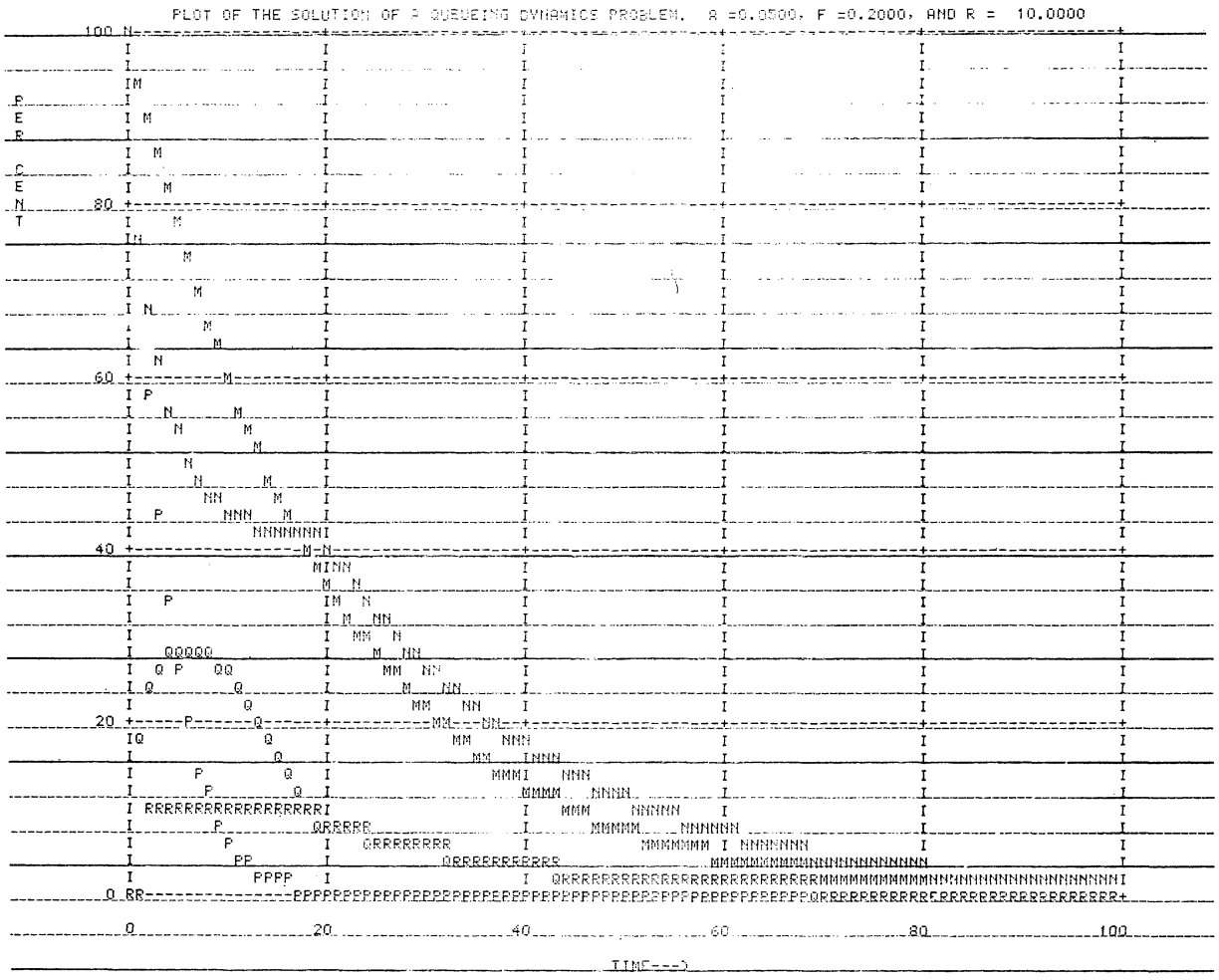
```
$COMPILE MAD, PUNCH OBJECT                                    QUE  00
R  N IS THE NUMBER AVAILABLE, Q THE NUMBER AWAITING SERVICE,,
RR THE NUMBER SERVECED. A IS THE ATTRITION HAZARD, F THE FAILU
RRE HAZARD.  N,Q, AND R DIMENSIONED TMAX.  TMAX INTEGER

 EXTERNAL FUNCTION (N,Q,R,M,P,A,F,RMAX,TMAX, TF)
 INTEGER T, TMAX
 ENTRY TO QUE1.
 TF(0) = 0
 M(0) =100.
 P(0) =100.
 N(0) = 100.
 Q(0) = 0.
 R(0) = 0.
 THROUGH ALPHA, FOR T=1,1,T.G.TMAX
 TF(T) = T
 WHENEVER Q(T-1).G.RMAX
 R(T) = RMAX
 OTHERWISE
 R(T) = Q(T-1)
 END OF CONDITIONAL
 M(T) = (1.-A)*M(T-1)
 P(T) = (1.-A)*(1.-F)*P(T-1)
 N(T) = R(T) + (1.-A)*(1.-F)*N(T-1)
ALPHA   Q(T) = Q(T-1)+F*(1.-A)*N(T-1) - R(T)
 FUNCTION RETURN
 END OF FUNCTION
```

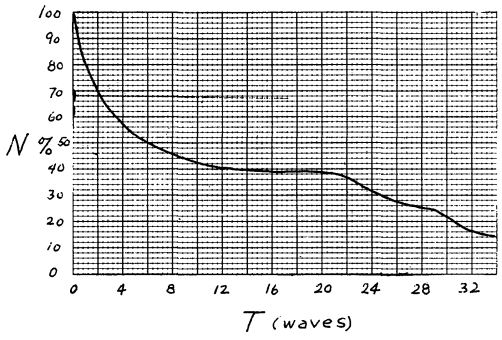
Computer Output and Results

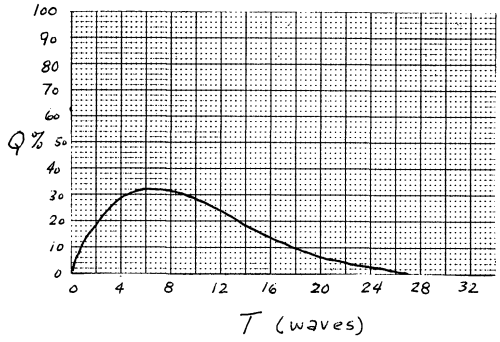
Samples of the digital and analog computer outputs are shown below.

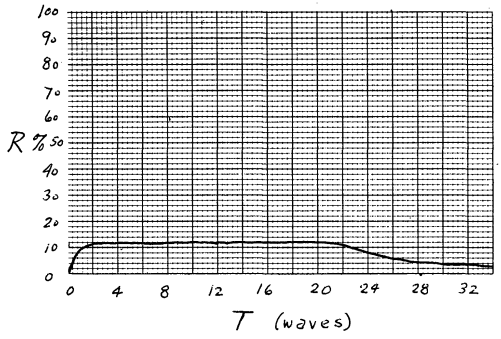
Digital Computer Output



Analog Computer Output







Discussion of Results

The output of the digital computer in the case of this problem is simply thirty-six (3x4x3) one-page plots of the quantities M, N, P, Q, and R against time (T) for 100 waves. Since the problem is intended to show the qualitative behavior of the system as a function of the parameters, the precision obtainable from the plots is sufficient.

In the analog computer output each heavy horizontal division represents two waves; each heavy vertical division represents ten percent.

Instructor's Comments and Critique

As stated in Section I, the problem has been selected so that it will provide a simple example of computer applications. In a practical case, it would be desirable to bring in much greater realism. One way in which this could be accomplished would be to let the problem parameters (the attrition rate, for example) vary with time as they almost certainly would in reality. Another way to provide added realism is to treat the problem stochastically by bringing in the probability distributions of attrition, failure, and repair times; this course would, however, require a different method of computer solution, namely, Monte Carlo simulation.

A Queueing Dynamics Problem

The analog portion of this presentation has actually been used in a course on analog computer applications. It was assigned as a lab project to three students. Since one of these students was in NROTC, there was no problem with face validity. The analog output was transcribed from a curve in their lab report.

The computer representations of this problem can also be used to represent problems in other areas. Consider, for example, the order-processing activity in a large mail order house. This activity may be considered to be a "black box" for purposes of system design. With given inputs, i.e., incoming orders, as a function of time, the outputs, filled orders, can be determined if the transfer function of the black box is known. This transfer function is obtained by simulating the system and applying a unit impulse, which is the equivalent of the conditions of this problem. Here A represents the daily rate of filling orders, F the percentage of incoming orders requiring special handling, and R the rate of processing these special handling orders and returning them to the regular order-filling channels.

A somewhat facetious situation involving traffic also is represented by the conditions of this problem. A company employs 100 people who, at quitting time, all attempt to leave the company parking lot at the same time. The parking lot exit is at an intersection controlled by a traffic light, and on the corner is a tavern. The light, traffic controlled, lets A percent of the waiting cars through at each change; F percent of the waiting drivers get disgusted and go into the tavern for a beer. However, the service rate at the bar, R, is restricted to a maximum value, R_{max}. Find the time required to empty the parking lot and the length of the queue, Q, at the bar.

Example Problem No. 58

OPTIMIZING MACHINE LOADING

by

Edward T. Kirkpatrick

Mechanical Engineering Department

The University of Toledo

Course: Engineering Analysis Credit Hours: 3 Level: Senior

Statement of the Problem

A situation is proposed where there is a list of jobs to be done giving the earliest possible starting time, latest possible finishing time and the running time for each job. It is known that the lowest cost results by doing as much work as possible on Machine 1, and it is assumed that other machines are available to do the jobs left over. The problem is to develop a computer program to list the jobs and their starting times so that Machine 1 has maximum loading.

It is suggested that the program be tested with a list of jobs whose running times total twenty four-hours, and that the Machine 1 running time be eight hours.

Solution

For the purposes of this problem, it will be assumed that maximum loading will occur if the jobs are fitted timewise in ascending order of job size, provided they are ready to be started.

Since the original job list could be quite random, the SHARE library subroutine generating random numbers was used to establish the job starting times and running times. Because the random number generator produces an eight decimal digit number between zero and one, and the starting times were to be between zero and eight hours, the random number was multiplied by 80, put into integer form and then divided by ten to give a number with one decimal between zero and eight. A similiar procedure was used to generate a running time between zero and three with one decimal. Number generation continued until the sum of job sizes exceeded 24 hours. Thus a three column array was generated giving job number, running time and earliest possible starting time.

In order to arrange the jobs in ascending order of size, the job size column (col. 2) was loaded into a sorting column (col. 5) and sorted in ascending order along with the associated job numbers and starting times (cols. 4 and 6).

The process of scanning the jobs to see what jobs were ready to start was initiated by comparing the job start times with zero time in descending order of job size until a job was found. Jobs of zero running time were discarded. If no job was found in the total list of jobs, the time was incremented by 0.1 hours, and the search repeated.

When a job was found with starting time equal to or less than the time under consideration, it had to be checked to see if it could be finished before the end of the shift by adding its running time to the "present" time. A check was also made to see if the job would be finished before the latest possible finishing time which was arbitrarily taken as being the earliest possible starting time plus three times the running time. If all these tests were passed, the job was added to the job schedule list together with its job number and actual computed starting time (cols. 7,8 and 9). So that a "left-over" job list could be easily recognized, a digit 8 was placed in the starting time column of each job successfully scheduled.

A new time was calculated by adding the running time of the last job and the scan was repeated and jobs shifted within their allowable ranges until the counter on time reached a maximum of eight hours.

A list of jobs left over was then made in descending order of job size by looking at starting times (col. 6) and if an eight was not present, the job was listed as being left over.

In order to check on the loading, a sum of the jobs "fitted" within the right hours was made, and then the whole array, and job hour sum were printed. It is then assumed that this information (for the real case) would then be given to management or a machine room supervisor, who would then handle the jobs as listed, and make arrangements for the jobs left over. For purposes of illustration three cases were run.

Table of Symbols

A(I,1) = Job Number

A(I,2) = Job Running Time $0 \leq A(I,2) \leq 3$

A(I,3) = Job Starting Time $0 \leq A(I,3) \leq 8$

SUM = Total Running Time of All Jobs

A(I,4) = Job Numer
A(I,5) = Job running Time
A(I,6) = Job Starting Time

} Sorted in ascending values of running time

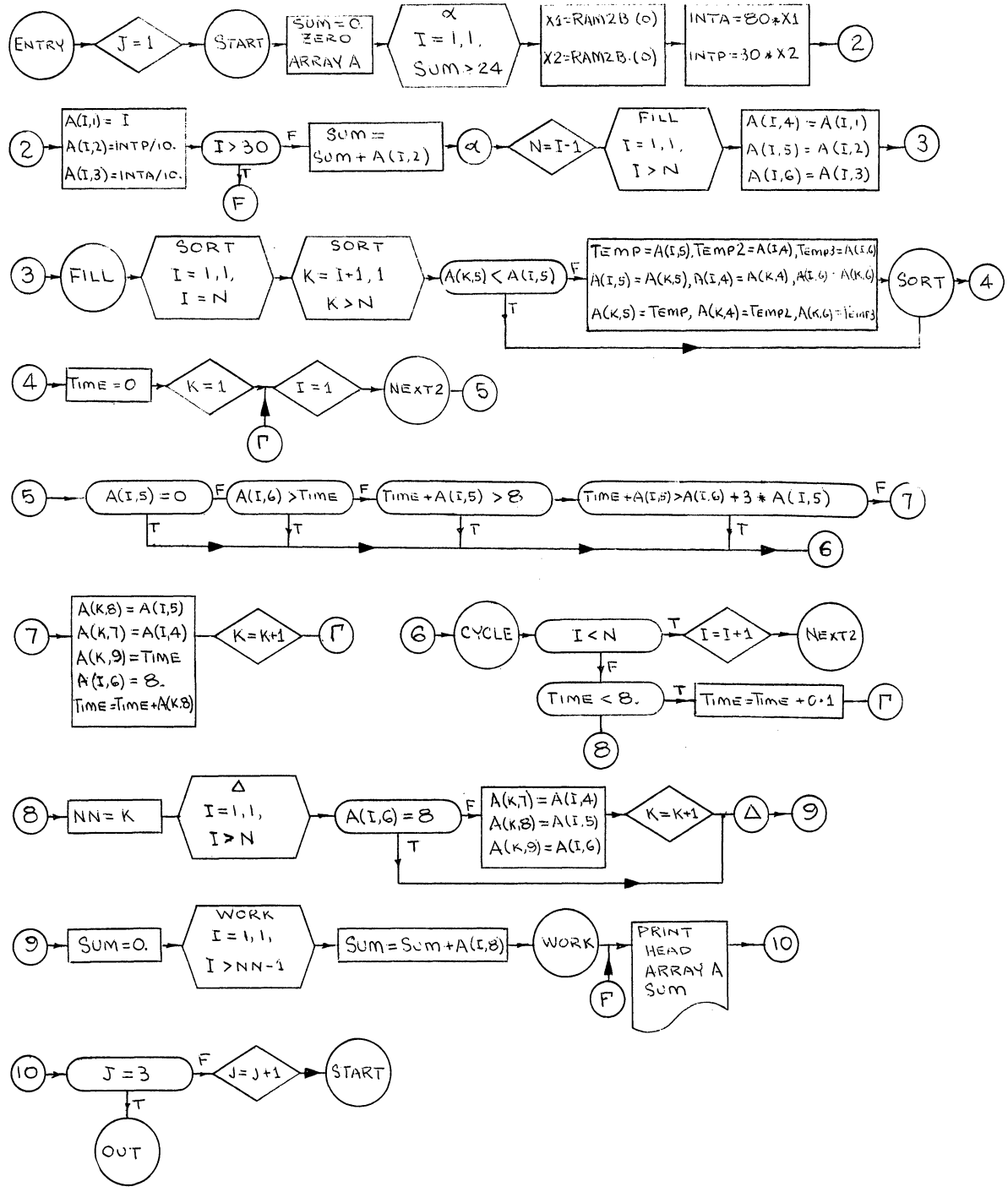
TIME = Time During Shift

A(K,7) = Job Number of K-th Assigned Job

A(K,8) = Running Time of K-th Assigned Job

A(K,9) = Starting Time of K-th Assigned Job

Flow Diagram



MAD Program

```
 E.T.KIRKPATRICK              D002N  1            003   010   150      OPTL0000
$ COMPILE MAD,PRINT OBJECT, DUMP, EXECUTE
          R***THIS PROGRAM USES THE RANDOM NUMBER GENERATOR TO ASSIGN
          RSTARTING TIMES AND JOB LENGTHS UNTIL TOTAL JOB LOAD EQUALS
          RTWENTY FOUR HOURS.THE JOBS ARE THEN SORTED IN DESCENDING
          RORDER ACCORDING TO JOB SIZE.THE PROGRAM THEN SETS UP A
          RJOB SCHEDULE WHICH LOADS IN AN OPTIMUM FASHION AND LISTS BOTH
          RTHE OPTIMUM JOB ASSIGNMENT AND JOBS LEFT OVER.
          R                    BY
          R              *** E T KIRKPATRICK ***
           DIMENSION A(300,ADIM)
           VECTOR VALUES ADIM=2,1,10
           INTEGER I,INTA,INTP,N,NN,K ,J
           J=1
START      SUM=0.
           EXECUTE SPRAY.(  0      ,A(1,1)...A(20,10))
           THROUGH ALPHA, FOR I=1,1,SUM.GE.24.
           X1=RAM2B.(0)
           X2=RAM2B.(0)
           INTA=80*X1
           INTP=30*X2
           A(I,1)=I
           A(I,2)=INTP/10.
           A(I,3)=INTA/10.
           WHENEVER      I.G.30, TRANSFER TO FAULT
ALPHA      SUM=SUM+A(I,2)
           N=I-1
          RSORT IN DESCENDING ORDER
           THROUGH FILL, FORI=1,1,I.G.N
           A(I,4)=A(I,1)
           A(I,6)=A(I,3)
FILL       A(I,5)=A(I,2)
           THROUGH SORT, FOR I=1,1,I.E.N
           THROUGH SORT, FOR K=I+1,1,K.G.N
           WHENEVER A(K,5).L.A(I,5),TRANSFER TO SORT
           TEMP=A(I,5)
           A(I,5)=A(K,5)
           A(K,5)=TEMP
           TEMP2=A(I,4)
           A(I,4)=A(K,4)
           A(K,4)=TEMP2
           TEMP3 =A(I,6)
           A(I,6)=A(K,6)
           A(K,6)=TEMP3
SORT       CONTINUE
          RASSIGN  JOB  LOADING
           TIME=0.
           K=1
LAMBDA     I=1
NEXT2      WHENEVER A(I,5).E.0., TRANSFER TO CYCLE
           WHENEVER A(I,6).G.(TIME+ .001), TRANSFER TO CYCLE
           WHENEVER TIME +A(I,5).G.8., TRANSFER TO CYCLE
           WHENEVER TIME+A(I,5).G.A(I,6)+3.*A(I,5),TRANSFER TO CYCLE
           A(K,8)=A(I,5)
           A(K,7)=A(I,4)
           A(K,9)=TIME
           A(I,6)=8.
           TIME=TIME+A(K,8)
           K=K+1
           WHENEVER K.G.N, TRANSFER TO FAULT
           TRANSFER TO LAMBDA
```

MAD Program (continued)

```
CYCLE       WHENEVER I.L.N
            I=I+1
            TRANSFER TO NEXT2
            OR WHENEVER TIME.L.8.
            TIME=TIME+0.1
            TRANSFER TO LAMBDA
            OTHERWISE
            TRANSFER TO NEXT
            END OF CONDITIONAL
           RLIST JOBS LEFT OVER
NEXT        NN=K
            THROUGH DELTA, FOR I=1,1,I.G.N
            WHENEVER A(I,6).E.8.,TRANSFER TO DELTA
            A(K,7)=A(I,4)
            A(K,8)=A(I,5)
            A(K,9)=A(I,6)
            K=K+1
DELTA       CONTINUE
           RCHECK SUM
            SUM=0.
            THROUGH WORK,FOR I=1,1,I.G.NN-1
WORK        SUM=SUM+A(I,8)
FAULT       CONTINUE
            PRINT FORMAT HEAD1
            PRINT FORMAT OUT4,A(1,1)...A(N,10)
            PRINT FORMAT OUT5,$TOTAL LOAD IN HOURS IS$,SUM
           RFORMATS
            VECTOR VALUES HEAD1=$1H2,32HLISTING OF JOBS AND SCHEDULING
           1S7,10HTOTAL LOAD,S15,8HORDERING,S16,8HSCHEDULE/S6,3HJOB,S4,4H
           2SIZE,S3,5HSTART,S5,3HJOB,S4,4HSIZE,S3,5HSTART,S5,3HJOB,S4,4HS
           3IZE,S4,5HSTART,S2,7HCOMMENT*$
            VECTOR VALUES OUT4 =$1H , 9F8.1,S6,1F2.1*$
            VECTOR VALUES OUT5=$1HO,4C6,F8.2*$
FINI        CONTINUE
            WHENEVER J.E.3, TRANSFER TO OUT
            J=J+1
            TRANSFER TO START
OUT         END OF PROGRAM
```

Computer Output

LISTING OF JOBS AND SCHEDULING

TOTAL LOAD			ORDERING			SCHEDULE			
JOB	SIZE	START	JOB	SIZE	START	JOB	SIZE	START	COMM
1.0	2.5	0.8	14.0	2.9	8.0	14.0	2.9	0.6	.
2.0	2.5	4.7	10.0	2.7	6.8	1.0	2.5	3.5	.
3.0	1.7	2.7	2.0	2.5	4.7	17.0	1.9	6.0	.
4.0	0.1	5.1	1.0	2.5	8.0	10.0	2.7	6.8	.
5.0	1.9	3.0	9.0	2.4	2.4	2.0	2.5	4.7	.
6.0	0.1	2.7	12.0	2.2	1.9	9.0	2.4	2.4	.
7.0	0.5	0.7	17.0	1.9	8.0	12.0	2.2	1.9	.
8.0	0.2	7.4	5.0	1.9	3.0	5.0	1.9	3.0	.
9.0	2.4	2.4	3.0	1.7	2.7	3.0	1.7	2.7	.
10.0	2.7	6.8	13.0	1.3	5.7	13.0	1.3	5.7	.
11.0	0.4	5.2	16.0	1.0	7.2	16.0	1.0	7.2	.
12.0	2.2	1.9	7.0	0.5	0.7	7.0	0.5	0.7	.
13.0	1.3	5.7	11.0	0.4	5.2	11.0	0.4	5.2	.
14.0	2.9	0.6	8.0	0.2	7.4	8.0	0.2	7.4	.
15.0	0.0	2.1	6.0	0.1	2.7	6.0	0.1	2.7	.
16.0	1.0	7.2	4.0	0.1	5.1	4.0	0.1	5.1	.
17.0	1.9	5.8	15.0	0.0	2.1	15.0	0.0	2.1	.

TOTAL LOAD IN HOURS IS 7.30

Computer Output (continued)

LISTING OF JOBS AND SCHEDULING

TOTAL LOAD			ORDERING			SCHEDULE			
JOB	SIZE	START	JOB	SIZE	START	JOB	SIZE	START	COMM
1.0	2.5	0.9	14.0	2.7	4.7	11.0	1.8	0.2	.
2.0	0.5	4.2	16.0	2.6	7.5	1.0	2.5	2.0	.
3.0	1.8	4.4	1.0	2.5	8.0	3.0	1.8	4.5	.
4.0	0.5	2.6	15.0	2.1	5.4	13.0	1.7	6.3	.
5.0	2.0	0.3	5.0	2.0	0.3	14.0	2.7	4.7	.
6.0	1.2	2.5	12.0	1.9	6.6	16.0	2.6	7.5	.
7.0	0.3	2.9	11.0	1.8	8.0	15.0	2.1	5.4	.
8.0	0.7	3.9	3.0	1.8	8.0	5.0	2.0	0.3	.
9.0	1.4	1.4	13.0	1.7	8.0	12.0	1.9	6.6	.
10.0	0.7	6.4	9.0	1.4	1.4	9.0	1.4	1.4	.
11.0	1.8	0.2	6.0	1.2	2.5	6.0	1.2	2.5	.
12.0	1.9	6.6	10.0	0.7	6.4	10.0	0.7	6.4	.
13.0	1.7	5.3	8.0	0.7	3.9	8.0	0.7	3.9	.
14.0	2.7	4.7	4.0	0.5	2.6	4.0	0.5	2.6	.
15.0	2.1	5.4	2.0	0.5	4.2	2.0	0.5	4.2	.
16.0	2.6	7.5	7.0	0.3	2.9	7.0	0.3	2.9	.
TOTAL LOAD IN HOURS IS			7.80						

LISTING OF JOBS AND SCHEDULING

TOTAL LOAD			ORDERING			SCHEDULE			
JOB	SIZE	START	JOB	SIZE	START	JOB	SIZE	START	COMM
1.0	2.0	2.9	16.0	2.9	7.4	17.0	1.9	0.3	.
2.0	0.7	7.6	15.0	2.8	8.0	6.0	2.8	2.2	.
3.0	0.4	4.4	9.0	2.8	2.8	15.0	2.8	5.0	.
4.0	2.5	2.5	6.0	2.8	8.0	16.0	2.9	7.4	.
5.0	0.9	0.9	4.0	2.5	2.5	9.0	2.8	2.8	.
6.0	2.8	1.0	1.0	2.0	2.9	4.0	2.5	2.5	.
7.0	0.2	4.6	17.0	1.9	8.0	1.0	2.0	2.9	.
8.0	0.8	5.9	14.0	1.8	2.2	14.0	1.8	2.2	.
9.0	2.8	2.8	11.0	1.7	6.1	11.0	1.7	6.1	.
10.0	0.2	5.8	13.0	1.1	0.4	13.0	1.1	0.4	.
11.0	1.7	6.1	5.0	0.9	0.9	5.0	0.9	0.9	.
12.0	0.2	3.9	8.0	0.8	5.9	8.0	0.8	5.9	.
13.0	1.1	0.4	2.0	0.7	7.6	2.0	0.7	7.6	.
14.0	1.8	2.2	3.0	0.4	4.4	3.0	0.4	4.4	.
15.0	2.8	3.6	12.0	0.2	3.9	12.0	0.2	3.9	.
16.0	2.9	7.4	10.0	0.2	5.8	10.0	0.2	5.8	.
17.0	1.9	0.3	7.0	0.2	4.6	7.0	0.2	4.6	.
TOTAL LOAD IN HOURS IS			7.50						

Discussion of Results

The program output shows the results of three cases generated by the program itself. Comparison of the columns under the headings "Total Load", "Ordering", and "Schedule" reveals that the data processing problem of fitting the job load using the method proposed has been accomplished in each case.

It is interesting to note that the only "lost" time in each case was at the beginning and end of the scheduled eight hours. This undoubtedly occurs because of the wide selection of jobs available. It is possible that for a real set of data from an industrial or commercial source, the job sizes and start times would not be as randomly distributed. The resulting fit of the jobs would then not be as good as the examples shown.

This problem could possibly be used to introduce the engineering student to data processing. The topic is of engineering interest because it deals with machines, but the decisions are simple and readily programmed on a computer. The fact that the computer is used to generate its own trial data undoubtedly will be of interest to students.

Example Problem No. 59

REGRESSION ANALYSIS

by

J. T. Elrod
Department of Industrial Engineering
University of Houston

Course: Work Measurements Credit Hours: 3 Level: Junior

Statement of Problem

Write a MAD program which uses regression equations and produces an equation for the data below.

Oxyacetylene-Torch Cutting*

Metal Thickness, Inches	Time, Minutes to Cut 1.0 Inch
1/4	.036
3/8	.037
1/2	.039
3/4	.042
1	.046
1 1/4	.050
1 1/2	.053
2	.059
2 1/2	.065
3	.072
3 1/2	.077
4	.084
4 1/2	.091
5	.100
5 1/2	.106
6	.111
6 1/2	.118
7	.125
7 1/2	.134
8	.143

Instructor's Solution

Development of standard data and formulas for synthesizing operation standard times is an important industrial engineering function. This problem represents a very simple situation where the data is essentially a straight-line function with one independent and one dependent variable. Either least squares or regression equations can be used. The solution shown is an extension of the problem as assigned, including a calculation of the standard error of estimate. Although such additional statistics were not required, some of the students developed a number of such values (based on material from other courses). A further extension of the problem would be to use a general type solution with polynomial and linear equations subroutines. However, time was limited and the simple solution was selected. The two parameters sought were y-axis intercept, A, and slope, B. The regression equations for these are:

(*) The data for this problem were obtained from W. A. Nordhoff, Machine-Shop Estimating, New York: McGraw-Hill Book Company, Inc. 1947, p. 452.

$$A = \frac{\Sigma(x^2)(\Sigma y) - (\Sigma x)(\Sigma xy)}{N(\Sigma x^2) - (\Sigma x)(\Sigma y)}$$

$$B = \frac{N(\Sigma xy) - (\Sigma x)(\Sigma y)}{N(\Sigma(x^2)) - (\Sigma x)(\Sigma y)}$$

and,

$$y = A + B\,x$$

The standard error of estimate (deviations about a given curve, rather than the mean) is given as,

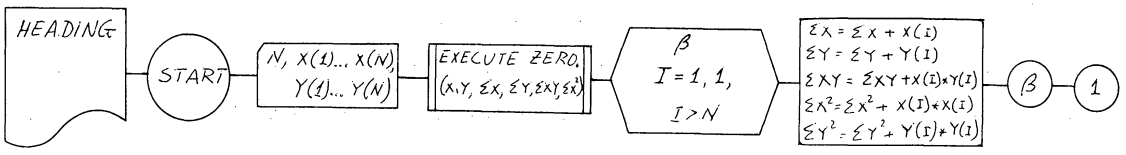
$$SE = \sqrt{\frac{\Sigma(D_i^2)}{N-2}}$$

where $\Sigma(D_i^2)$ is the sum of the squares of the differences (vertically) of the observed values (y_i) from the equation values, (YT_i).

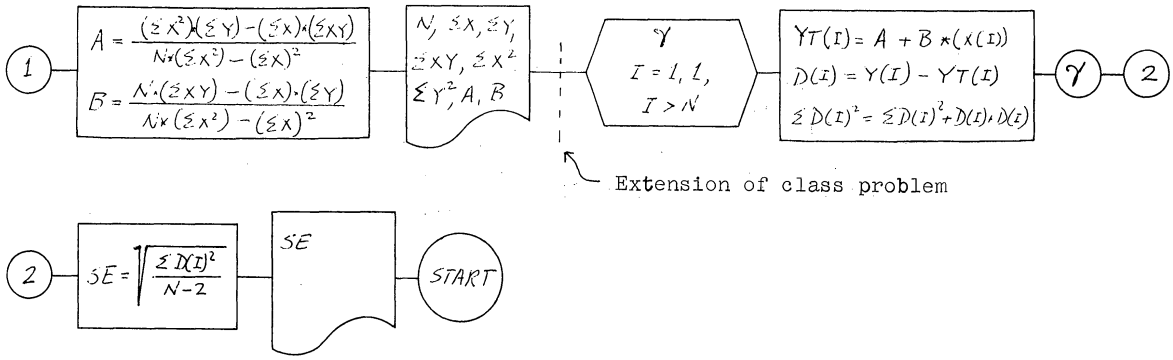
Table of Symbols

Flow Chart	Program	Definition
X	X	Thickness of metal, inches
Y	Y	Time, minutes
ΣX	SUMX	Sum of the x_i
ΣY	SUMY	Sum of the y_i
ΣX^2	SUMXSQ	Sum of the x_i^2
ΣY^2	SUMYSQ	Sum of the y_i^2
ΣXY	SUMXY	Sum of the $x_i y_i$
ΣD^2	SUMDSQ	Sum of the D_i^2
A	A	y-axis intercept (MAD)
A	C	y-axis intercept (FORTRAN)
YT	YT	Calculated values of y
D(I)	D(I)	Difference between calculated and observed values of y
SE	STDERR	Standard error of estimate

Flow Diagram



Flow Diagram, Continued



MAD Program and Data

```
J T ELROD                          X56-N  6          001   015   015
*COMPILE MAD, EXECUTE, DUMP
            PRINT FORMAT HDNG
            PUNCH FORMAT HDNG
START       READ FORMAT METAL, N, X(1)...X(N)
            READ FORMAT TIME,N,Y(1)...Y(N)
            EXECUTE ZERO.(X,Y,SUMX,SUMY,SUMXY,SUMXSQ,SUMYSQ,SUMDSQ)
            THROUGH BETA, FOR I=1, 1, I.G.N
            SUMX=SUMX +X(I)
            SUMY=SUMY + Y(I)
            SUMXY=SUMXY +(X(I)*Y(I))
            SUMXSQ=SUMXSQ +(X(I)*X(I))
BETA        SUMYSQ=SUMYSQ +(Y(I)*Y(I))
            PRINT FORMAT SUMS, N,  SUMX,SUMY,SUMXY,SUMXSQ,SUMYSQ
            A = ((SUMXSQ*SUMY)-(SUMX*SUMXY))/((N*SUMXSQ)-(SUMX*SUMX))
            B=((N*SUMXY)-(SUMX*SUMY))/((N*SUMXSQ)-(SUMX*SUMX))
            PRINT FORMAT R, A, B
            THROUGH GAMMA, FOR I=1, 1, I.G.N
            YT(I) =  A  +  (B*X(I))
            D(I)  = (Y(I)  -  YT(I))
GAMMA       SUMDSQ  =  SUMDSQ + (D(I) *D(I))
            STDERR = SQRT.(SUMDSQ/(N-2.))
            PRINT FORMAT STD, STDERR
            VECTOR VALUES STD = $1H0, S4,  30HSTANDARD ERROR OF ESTIMATE
           1=    F10.6*$
            TRANSFER TO START
            INTEGER I, N
            DIMENSION X(20),Y(20) , YT(20),  D(20)
            VECTOR VALUES HDNG=$1H1,S30,19HREGRESSION EQUATION//
           1S18,43HTIME (MINUTES PER LINEAR INCH) TO FLAME-CUT//
           2S23,35HVARIOUS THICKNESSES OF BOILER PLATE *$
            VECTOR VALUES METAL = $I3/(10F6.3)*$
            VECTOR VALUES TIME  = $I3/(10F6.3)*$
            VECTOR VALUES SUMS =$1H0,S5,3HN =I2,S10,13HSUM OF X(I) =
           1F8.3,S10,13HSUM OF Y(I) = F8.3, // S6,17HSUM OF X(I)Y(I) =
           2F10.6, S10,21HSUM OF X(I) SQUARED = F10.6, // S6, 21HSUM OF Y
           3(I) SQUARED =  F10.6 ///*$
            VECTOR VALUES R=$S6,4HY =  F10.6,S2,3H+  F10.6,1HX    *$
            END OF PROGRAM
*DATA
 20
 .250  .375  .500  .750 1.000 1.250 1.500 2.000 2.500 3.000
 3.500 4.000 4.500 5.000 5.500 6.000 6.500 7.000 7.500 8.000
 20
  .036  .037  .039  .042  .046  .050  .053  .059  .065  .072
  .077  .084  .091  .100  .106  .111  .118  .125  .134  .143
```

Fortran Program and Data

```
J T ELROD                          X56-N 6F          001   010   000
*COMPILE FORTRAN, EXECUTE, DUMP
      DIMENSION A(20, 2)
      WRITE OUTPUT TAPE 6, 5
    5 FORMAT  (1H1, 30X, 19HREGRESSION EQUATION// 18X,
     143HTIME (MINUTES PER LINEAR INCH) TO FLAME-CUT // 18X,
     235HVARIOUS THICKNESSES OF BOILER PLATE    )
   10 READ INPUT TAPE 7, 20,    ((A(I,J),I=1,20),J=1,2)
      READ INPUT TAPE 7,  21, N
      SUMX = 0
      SUMY = 0
      SUMXY = 0
      SUMXSQ = 0
      DO 15 I = 1, 20
      SUMX = SUMX + A(I,1)
      SUMY = SUMY + A(I, 2)
      SUMXY = SUMXY + (A(I,1) * A(I,2))
   15 SUMXSQ = SUMXSQ + (A(I,1) * A(I,1))
      Q = N
      C = ((SUMXSQ * SUMY) - (SUMX * SUMXY))/((Q*SUMXSQ)-(SUMX*SUMX))
      B = ((Q*SUMXY) - (SUMX*SUMY))/((Q*SUMXSQ)-(SUMX*SUMX))
      WRITE OUTPUT TAPE 6, 8, N, SUMX, SUMY, SUMXY, SUMXSQ
      WRITE OUTPUT TAPE 6, 9, C, B
      GO TO 10
   20 FORMAT     (10F6.3)
   21 FORMAT (I3)
    8 FORMAT (1H0, 5X, 3HN =  I2, 10X, 13HSUM OF X(I) =  F8.3, 10X,
     113HSUM OF Y(I) =   F8.3,// 6X, 17HSUM OF X(I)Y(I) =  F10.6, 10X,
     221HSUM OF X(I) SQUARED =  F10.6//)
    9 FORMAT (6X, 4HY =  F10.6, 2X, 3H+   F10.6, 1HX)
*DATA
 .250  .375  .500  .750 1.000 1.250 1.500 2.000 2.500 3.000
3.500 4.000 4.500 5.000 5.500 6.000 6.500 7.000 7.500 8.000
 .036  .037  .039  .042  .046  .050  .053  .059  .065  .072
 .077  .084  .091  .100  .106  .111  .118  .125  .134  .143
20
```

Computer Results

REGRESSION EQUATION

TIME (MINUTES PER LINEAR INCH) TO FLAME-CUT

VARIOUS THICKNESSES OF BOILER PLATE

N =20 SUM OF X(I) = 70.625 SUM OF Y(I) = 1.588

SUM OF X(I)Y(I) = 7.313375 SUM OF X(I) SQUARED =376.328125

SUM OF Y(I) SQUARED = 0.149042

Y = 0.031947 + 0.013438X

STANDARD ERROR OF ESTIMATE = 0.001348

Discussion of Results

The print-out is in the form of an equation

$$Y = 0.031947 + 0.013438X$$

with results in minutes. Obviously, the values shown are not accurate past the third place (see original data), but are carried out to allow for rounding in actual use. The standard error of estimate is also in minutes, and indicates a very close approximation of the data to a straight line.

Instructor's Comments and Critique

The problem is a typical one in curve fitting, but actually too simple. A more sophisticated computer solution to a higher order curve would be more appropriate. The actual programming was similar in nature to another problem which was assigned earlier in the semester (see problem 51, p. 236-244 of ref. 12) so that not too much time was spent by the students in actual programming. As instructors and students alike come to this course with computer background, the programming can be accelerated to allow for more advanced solutions. This problem did help to firm up the student's grasp of programming and the nature of curve fitting (regression analysis). Actually, if the problem had been much more complex, under the circumstances the students would have been hard pressed for time. This problem did not detract materially, time-wise, from the course, but added another facet to an important area of analysis.

The students reacted favorably to this problem, again in some cases, carrying the problem past the assigned scope. They were beginning to feel "at home" in programming, and this confidence did a lot to leave a positive attitude toward computers and programming in their minds.

This problem was taken in stride, and although not the most efficient way to solve a rather simple problem, the computer seemed not an unnatural way. One or more problems of this nature would have been solved in the course anyway, so this time the computer was the means of solution.

In addition to the extensions referred to above, the instructor obtained a multiple linear regression program from the Computing Center, and ran a problem with data having one dependent and four independent variables. (This program, known as "Regression Analysis Program - Adapted from GM Program," can handle as many as sixty variables and an unlimited number of data items.) The results of this run were presented to the class as an example of the complex programs that can be handled.

Both the computer and the language are suitable, although a smaller machine and a language such as FORTRAN would have been more than adequate for this course.

The turn-around time became excessive and it was necessary to grant students extension of time for completion of the problem.

The general reaction to this program certainly balances out on the positive side. Teaching the course was made much more interesting and challenging by using the computer. The problems to be selected in the future should be carefully chosen and more significant ones developed. In the future, when students will come into the course with a background in computers, statistics, mathematics, etc., the programming will come as a matter of course. It was somewhat forced in the present instance. Generally speaking, the experiment was a success.

Example Problem No. 60

CRITICAL PATH SCHEDULING

by

Richard C. Wilson

Industrial Engineering Department

University of Michigan

Course: Production Control Credit Hours: 2 Level: Senior

Statement of Problem

A pipeline replacement project consists of the following list of tasks, the sequence in which the tasks must be performed, and the estimated time for each task:

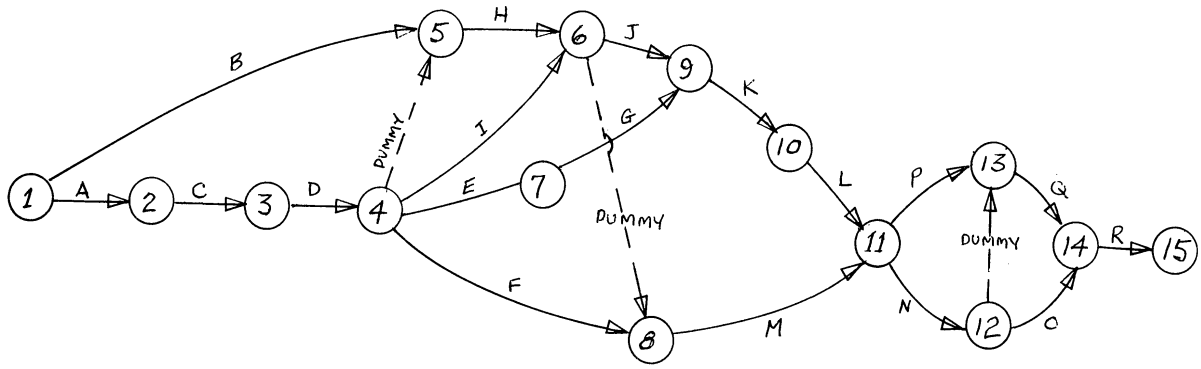
Task k		Sequence Code i,j	Elapsed Time (Days) dij
A	Lead Time	1,2	10
B	Line Available	1,5	28
C	Measure Sketch	2,3	2
D	Develop Material List	3,4	1
E	Procure Pipe	4,7	30
F	Procure Valves	4,8	45
G	Prefab Sections	7,9	5
H	Deactivate Line	5,6	1
I	Erect Scaffold	4,6	2
J	Remove Old Pipe	6,9	6
K	Place New Pipe	9,10	6
L	Weld Pipe	10,11	2
M	Place Valves	8,11	1
N	Fit Up	11,12	1
O	Pressure Test	12,14	1
P	Insulate	11,13	4
Q	Remove Scaffold	13,14	1
R	Clean Up	14,15	1

The figure below shows the network representation of the sequence restrictions constructed by considering each task q in turn, which tasks precede the task, which immediately follows it, and which can be done at the same time. The sequence code indicates the initial node of the tasks by i and the terminating node by j. Additional scheduling restrictions are imposed by the following:

1. The material list (task D) must be completed before the line is deactivated (task H).
2. The line must be deactivated (task H) before valves can be placed (task M).
3. Fit-up (task N) must be completed before the scaffold is removed (task Q).

Treat these as dummy tasks with zero elapsed time but with sequence codes as indicated on the figure.

1. What is the earliest possible completion time?
2. What are the latest starting times for each task that will assure project completion on time?
3. What tasks within the main project can float in the schedule, and to what degree?



Instructor's Solution

Solutions to problems of this kind are obtained by the critical path scheduling technique. (1) Each task or "link" k may be considered to be a uni-directional branch of a network with origin node i and terminal node j, with i and j selected such that $j > i$. The required elapsed time, dij, is the "value" of the link k. The algorithm finds the value of the path through the network which has the largest total value from the origin to the completion node. Supplemental information on float times is also obtained for the n-link network.

The program input variables are:

N = number of "links" in the project network

I(K) = identity of the initial node of the kth link

J(K) = identity of the terminating node of the kth link

TITL(K,1)...TITL(K,3) = alphabetic description of task k

DIJ(K) = elapsed time required for task k

TI(I(K)) represents the earliest possible starting time and TE(J(K)) the latest completion time for link k whose initial node is I(K). TI(0) and TE(0) are set equal to zero. The value of the critical (or minimum time) path TE(J(N)) and the earliest starting time of each link k is obtained through loop A. The network is searched in a forward direction, by incrementing k, to

find the maximum of the earliest starting times at each node J(K). The latest completion times are then found by a backward search through the network by decrementing k starting at the completion node. The iteration loop THROUGH B carries out this search. The latest possible starting time for link k is LS(K) = TE(J(K))-DIJ(K). The earliest completion time for link k is EF(K) = TI(I(K)) + DIJ(K). The total float of those links not in the critical path is defined as TF(K) = TE(J(K))-TI(I(K))-DIJ(K), and represents the amount of time that the startup of a task can be shifted without affecting the main chain. "Free float", the amount of time the startup of a job can be shifted without affecting any other job, is defined as FF(K) = TI(J(K))-TI(I(K))-DIJ(K) where TI(J(K)) is the earliest start time of the task whose ith node corresponds to the jth node of the k link. The iteration THROUGH D completes the calculation of these values.

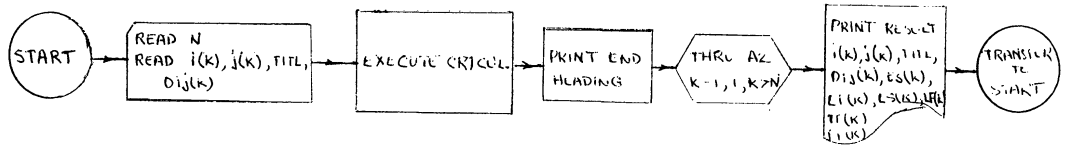
The solution program is written as an external function, in anticipation that its generality may make it desirable at future dates to incorporate the function in larger programs. Input data must be presented in specific arrangement:

1) I(K) < J(K) for all k
2) I(K) values must be non-decreasing
3) I(K) values must not omit any values in sequence from 1 to n-1

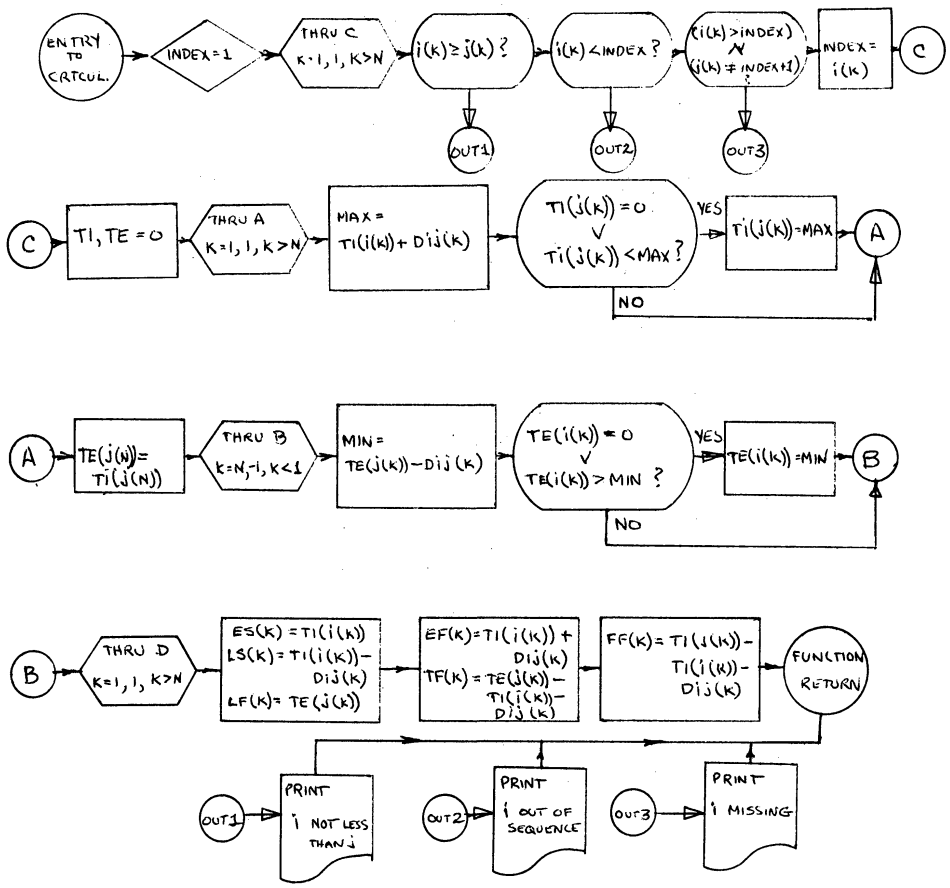
The three checks in iteration loop THROUGH C transfer to OUT1, OUT2, or OUT3 if one of these data requirements is violated and returns control to the main program.

The output array presents the possible task schedule times under the appropriate heading. The critical path is that sequence of tasks whose total float and free float are both zero. In this example, the sequence is therefore 1-2-3-4-8-11-13-14-15. The minimum time for project completion is the earliest finish time of the last task, 65 days in this example.

Flow Diagram for Main Program



Flow Diagram for Subroutine CRTCUL.



MAD Listing for Main Program

```
 R.C. WILSON                    D054N                  005   010   000       CRTCL
$COMPILE MAD,EXECUTE,DUMP,PRINT  OBJECT
          DIMENSION I(200),J(200),TITL(600,DIM),DIJ(200),ES(200),LS(20
         1),EF(200),LF(200),TF(200),FF(200),TI(200),TE(200)
          VECTOR VALUES DIM=2,1,3
          NORMAL MODE IS INTEGER
START      READ FORMAT ONE, N
          VECTOR VALUES ONE=$I10*$
          THROUGH A1, FOR K=1,1, K .G. N
          READ FORMAT ONE1, I(K),J(K),TITL(K,1)...TITL(K,3),DIJ(K)
          VECTOR VALUES ONE1=$2I5,S4,3C6,I3*$
A1        CONTINUE
          EXECUTE CRTCUL.(N,I,J,DIJ,ES,LS,EF,LF,TF,FF,TI,TE)
          PRINT FORMAT END
          VECTOR VALUES END=$79H1SEQUENCE  DESCRIPTION           JOB
         1    EARLIEST        LATEST          FLOAT  // 80H  CODE
         2                  DAYS     START*FINISH   START*FINISH   TOTA
         3L*FREE      //*$
          THROUGH A2, FOR K=1,1, K .G. N
A2        PRINT FORMAT RES,I(K),J(K),TITL(K,1)...TITL(K,3),DIJ(K),ES(K)
         1,EF(K),LS(K),LF(K),TF(K),FF(K)
          VECTOR VALUES RES=$2H  ,2I3,S4,3C6,3(I6,I9),I6*$
          TRANSFER TO START
          END OF PROGRAM
```

MAD Listing for Subroutine CRTCUL.

```
$COMPILE MAD,EXECUTE,DUMP,PRINT  OBJECT  ,PUNCH OBJECT          CRCL
          EXTERNAL FUNCTION (N,I,J,DIJ,ES,LS,EF,LF,TF,FF,TI,TE)
          ENTRY TO CRTCUL.
          NORMAL MODE IS INTEGER
          INDEX=1
          THROUGH C, FOR K=1,1, K .G. N
          WHENEVER I(K) .GE. J(K), TRANSFER TO OUT1
          WHENEVER I(K) .L. INDEX, TRANSFER TO OUT2
          WHENEVER I(K) .G. INDEX .AND. I(K) .NE. INDEX+1, TRANSFER TO
         1 OUT3
C         WHENEVER I(K) .E. INDEX+1, INDEX= I(K)
          EXECUTE ZERO.(TI,TE)
          THROUGH A,FOR K=1,1,K .G. N
          MAX=TI(I(K))+DIJ(K)
A         WHENEVER TI(J(K)) .E. 0 .OR. TI(J(K)) .L. MAX, TI(J(K))=MAX
          TE(J(N))=TI(J(N))
          THROUGH B, FOR K=N,-1, K .L. 1
          MIN=TE(J(K))-DIJ(K)
B         WHENEVER TE(I(K)) .E. 0 .OR. TE(I(K)) .G. MIN, TE(I(K))=MIN
          THROUGH D, FOR K=1,1, K .G. N
          ES(K)=TI(I(K))
          LS(K)=TE(J(K))-DIJ(K)
          EF(K)=TI(I(K))+DIJ(K)
          LF(K)=TE(J(K))
          TF(K)=TE(J(K))-TI(I(K))-DIJ(K)
D         FF(K)=TI(J(K))-TI(I(K))-DIJ(K)
          TRANSFER TO OUT
OUT1      PRINT FORMAT ERR1,K,K
OUT2      PRINT FORMAT ERR2,K
OUT3      PRINT FORMAT ERR3,K
          VECTOR VALUES ERR1=$1H1, 3H I(I3,18H) NOT LESS THAN J(I3,1H)*
         1$
          VECTOR VALUES ERR2=$1H1,3H I(I3,17H) OUT OF SEQUENCE*$
          VECTOR VALUES ERR3=$1H1, 3H I(I3,9H) MISSING *$
OUT       FUNCTION RETURN
          END OF FUNCTION
```

DATA for Main Program

```
$DATA
      21
 1    2    LEAD TIME            10
 1    5    LINE AVAILABLE       28
 2    3    MEASURE , SKETCH      2
 3    4    DEV. MAT,L LIST       1
 4    5    DUMMY                 0
 4    6    ERECT SCAFFOLD        2
 4    7    PROCURE PIPE         30
 4    8    PROCURE VALVES       45
 5    6    DEACTIVATE LINE       1
 6    8    DUMMY                 0
 6    9    REMOVE OLD PIPE       6
 7    9    PREFAB SECTIONS       5
 8   11    PLACE VALVES          1
 9   10    PLACE NEW PIPE        6
10   11    WELD PIPE             2
11   12    FIT UP                1
11   13    INSULATE              4
12   13    DUMMY                 0
12   14    PRESSURE TEST         1
13   14    REMOVE SCAFFOLD       1
14   15    CLEAN UP              1
```

Computer Output

SEQUENCE CODE		DESCRIPTION	JOB DAYS	EARLIEST START	EARLIEST FINISH	LATEST START	LATEST FINISH	FLOAT TOTAL	FLOAT FREE
1	2	LEAD TIME	10	0	10	0	10		
1	5	LINE AVAILABLE	28	0	28	16	44	16	
2	3	MEASURE , SKETCH	2	10	12	10	12		
3	4	DEV. MAT,L LIST	1	12	13	12	13		
4	5	DUMMY	0	13	13	44	44	31	1
4	6	ERECT SCAFFOLD	2	13	15	43	45	30	1
4	7	PROCURE PIPE	30	13	43	16	46	3	
4	8	PROCURE VALVES	45	13	58	13	58		
5	6	DEACTIVATE LINE	1	28	29	44	45	16	
6	8	DUMMY	0	29	29	58	58	29	2
6	9	REMOVE OLD PIPE	6	29	35	45	51	16	1
7	9	PREFAB SECTIONS	5	43	48	46	51	3	
8	11	PLACE VALVES	1	58	59	58	59		
9	10	PLACE NEW PIPE	6	48	54	51	57	3	
10	11	WELD PIPE	2	54	56	57	59	3	
11	12	FIT UP	1	59	60	62	63	3	
11	13	INSULATE	4	59	63	59	63		
12	13	DUMMY	0	60	60	63	63	3	
12	14	PRESSURE TEST	1	60	61	63	64	3	
13	14	REMOVE SCAFFOLD	1	63	64	63	64		
14	15	CLEAN UP	1	64	65	64	65		

Comments

The problem presented here is considered to be introductory to the concept of Critical Path Scheduling. Preparation of the program introduces the student to the "labeling" process typical of network algorithms. From two standpoints the problem is synthetic: the scope and complexity of the project is greatly reduced from real projects on which critical path scheduling is most useful; second, the full power of the technique is not exposed by the problem, since the cost minimization objective function has been omitted in the interest of simplicity. Programming such a parametric network flow problem, however, is not believed appropriate to the production control course because of time requirements. See problem 61, "Minimum Path Through a Network" for this extension.

Reference

Kelly, J. E. Jr., Sayer, J. S., Walker, M. R. "Critical Path Scheduling" Factory (July, 1960).

Example Problem No. 61

MINIMUM PATH THROUGH A NETWORK

by

Richard C. Wilson

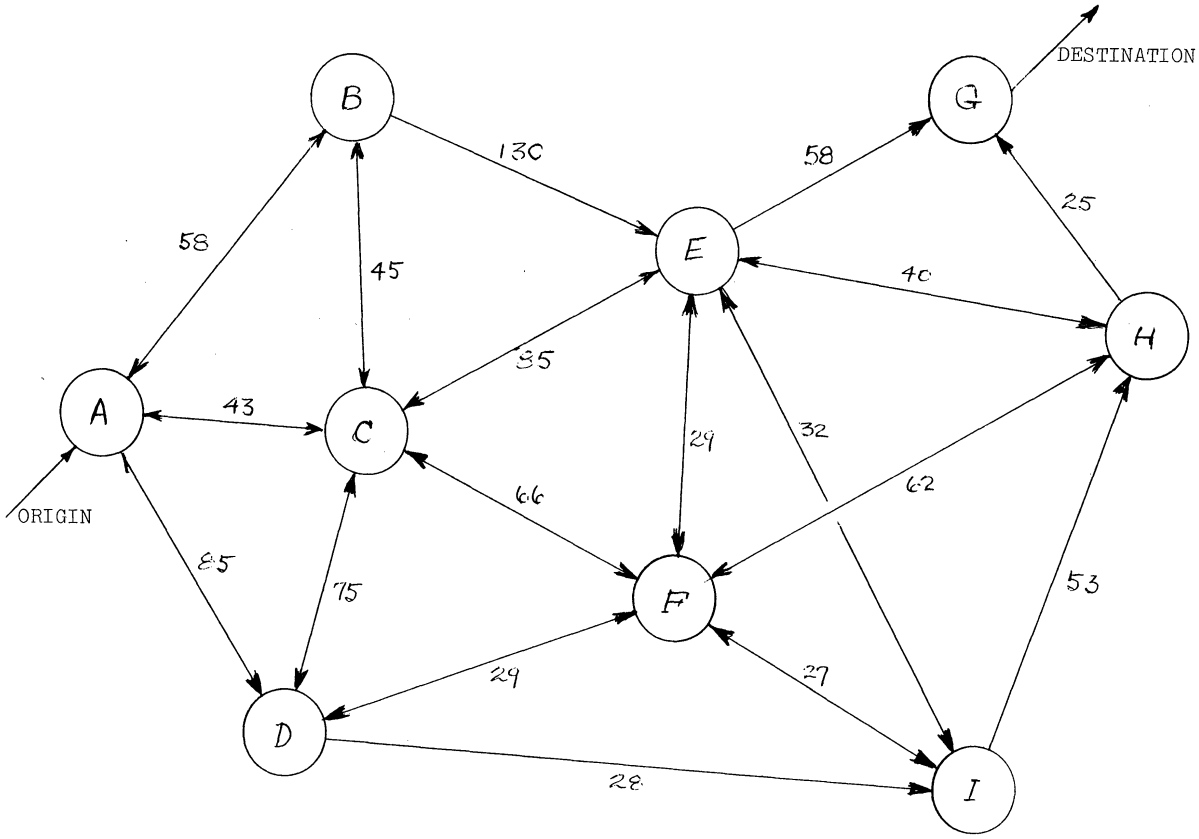
Industrial Engineering Department

University of Michigan

Course: Plant Flow Analysis Credit Hours: 3 Level: Graduate

Statement of Problem

A shipper at location A shown below has a carload of merchandise which is to be delivered to a customer at location G. A number of alternative transportation routes and intermediate junction points may be used for delivery. The shipper wishes to select a sequence of routes and junction points so that the total cost associated with the shipment is minimized. Cost for shipment between junction points are given in the figure below. Notice that shipment between some junctions can be made in only one direction. Find the sequence of shipping routes which has minimum total cost.



Example Problem No. 61

Solve the problem by preparing an external function in MAD which will be capable of handling similar network problems with as many as 500 one directional routes, or 250 two-directional routes. A main program for reading and printing the results should be prepared and the above problem used to check the program.

Instructor's Solution

Algorithms for solution to minimum path network problems are described by Moore* and Dantzig+. Briefly, the network consists of the junctions, or nodes, and the directed routes, or links, which connect the nodes.

The example problem therefore has 30 directional links, some of which connect the same two nodes except that they are in opposite directions. Associated with each directed link A is a value VN(A) which is the cost of shipment over link A. The thirty links of the given problem are arbitrarily assigned node identity numbers at each of the lettered junctions. The assignments are: A=2, B=1, C=3, D=4, E=5, F=6, G=8, H=9 and I=7. Thus, the link from node 2 to node 1 has a value of 58; the link from node 1 to node 2 also has a value of 58, etc. The solution then consists of devising an efficient method of examining all possible paths from the origin, SP, to the destination, EP, to determine which sequence of links has costs whose sum is a minimum.

The algorithm used is divided into two parts. A forward search through the network for the minimum value from the start point to each node, called the "minimum tree", is executed by statements labeled A1 to A3. After the destination is reached, the algorithm traces back through the minimum path found in order to identify the links in the path.

The following arguments must be furnished to the external function MINPAT.

SP = identity of the origin node

EP = identity of the destination node

ICN(A) = identity of the entering node to link A

FCN(A) = identity of the terminating node of link A

VN(A) = cost of link A

NODES = number of nodes in the network

NLNKS = number of links in the network

References:

*Moore, E. F. "The Shortest Path Through a Maze," presented at International Symposium on Theory of Switching, Havard University, 1957.

+Dantzig, G. B. "Discrete Variable Extremum Problems" Opns. Res. 5: p. 268 (1957).

The function MINPAT. returns:

CT	=	the number of links in the minimum path
OPTVAL	=	the value of the minimum path
HEAD(J)	=	identity of the entering node to jth link in the minimum path
TAIL(J)	=	identity of the terminating node of the jth link in minimum path
J	=	identity of link in minimum path starting with j=1 for final link counting backwards to j = CT for first link.

Four temporary storage arrays are used:

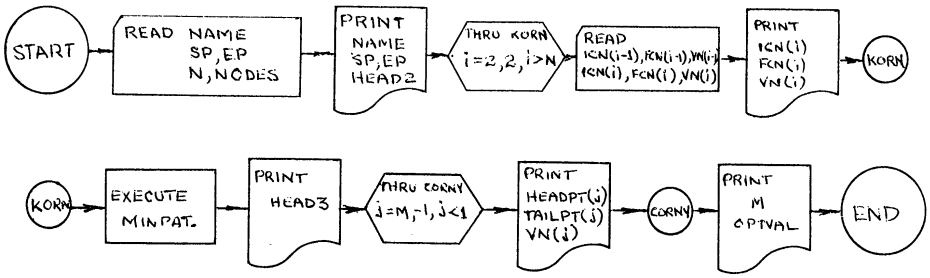
U(I)	=	cost of the tree to the current node i
Q(I)	=	a switch which is set equal to 0 after the node i has been "scanned"; otherwise Q(I) = 1
V(I)	=	identity of link whose terminating node is i
I(I)	=	identity of the initial node of link whose terminating node is i

To begin the algorithm, all values V(I) are initially set to ∞ ($\sim 10^{10}$) and all Q(I) to 0. The value of U(SP) at the starting node is set to 0, and Q(SP) to 1, indicating that the minimum tree to the starting node has value 0 and the node is to be scanned for all links which depart from it; i.e., all links whose ICN(A) = I = 1. The value of U(FCN(A)) then becomes the minimum of the value of the links VN(A) radiating from the SP node plus U(I), or the current value of U(FCN(A)). Q(FCN(A)) is set to 1 indicating that each of the new nodes is now to be scanned for all links which depart from it, that is, all links whose I(FCN(A)) = I. Q(SP) is set to 0, and the procedure iterated until all nodes have been scanned, and hence the minimum tree to each node evaluated. One of these nodes is of course the destination desired.

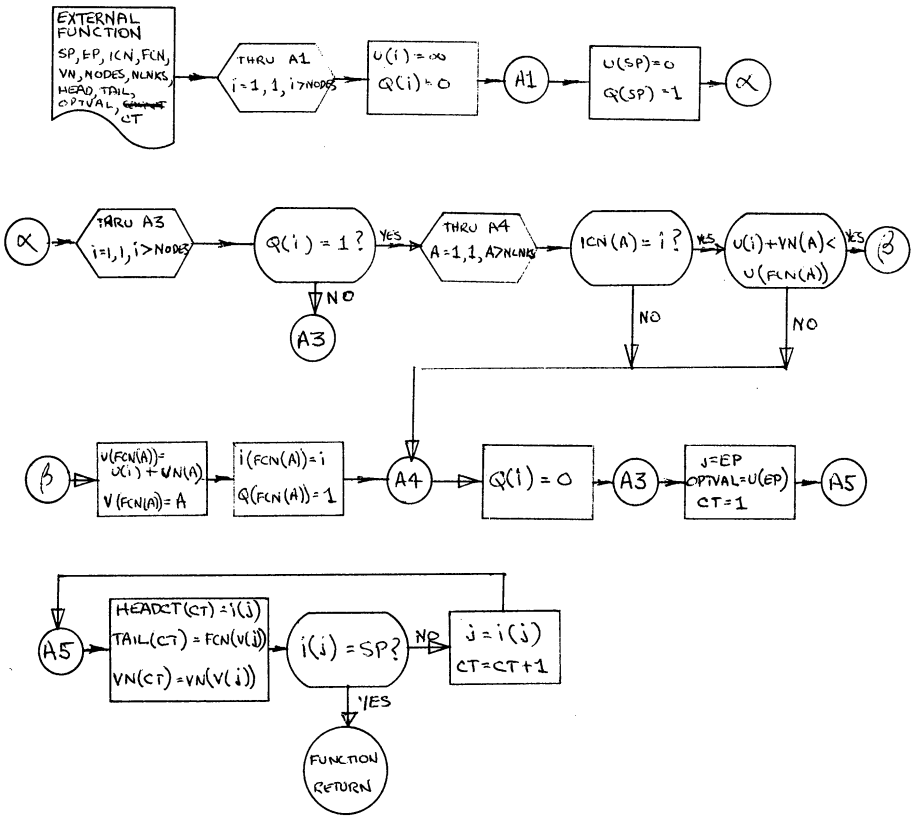
Beginning at A3, the algorithm executes a backward trace through the minimum path found by the method above. OPTVAL is set to U(EP), the counter CT to 1, HEAD(1) to I(EP), TAIL(1) to EP and VN(1) to VN(EP). The initial node of the link then is used as the final node of the preceding link on the minimum path and the process repeated until the starting point is reached. The counter CT then is equal to the number of links in the minimum path. Each link is identified by initial and terminating nodes stored in the HEAD and TAIL arrays.

The main calling program is used only to read in original data and for printing results. The external function therefore can be used in other programs to permit investigation of more advanced problems, such as stochastic or dynamic sequential network flow problems.

Flow Diagram for the Main Program



Flow Diagram for the Subroutine MINPAT.



MAD Listing of Main Program

```
 R.C. WILSON                    D054N              005   010   000
$COMPILE MAD,EXECUTE,DUMP,PRINT  OBJECT, PUNCH OBJECT                MNET
          DIMENSION ICN(200),HEADPT(200),TAILPT(200),FCN(200),VN(200),
         1NAME(12)
          NORMAL MODE IS INTEGER
START     READ FORMAT ONE1, NAME(1)...NAME(12)
          VECTOR VALUES ONE1=$12C6*$
          PRINT FORMAT ONE2,NAME(1)...NAME(12)
          VECTOR VALUES ONE2=$1H1, 12C6*$
          READ FORMAT ONE,SP,EP
          VECTOR VALUES ONE=$6I10*$
          PRINT FORMAT HEAD1,SP,EP
          VECTOR VALUES HEAD1=$1H0,S10,13H START POINT=I3,13H  END POIN
         1T= I3*$
          READ FORMAT ONE,N,NODES
          PRINT FORMAT HEAD2
          VECTOR VALUES HEAD2=$62H         (I)         (J)     IJ DIST.
         1  (I)        (J)  IJ DIST. *$
          THROUGH KORN, FOR I=2,2,I .G. N
          READ FORMAT ONE,ICN(I-1),FCN(I-1),VN(I-1),ICN(I),FCN(I),VN(I)
KORN      PRINT FORMAT ONE,ICN(I-1),FCN(I-1),VN(I-1),ICN(I),FCN(I),VN(I
         1)
          EXECUTE MINPAT.(SP,EP,ICN,FCN,VN,NODES,N,HEADPT,TAILPT,OPTVAL
         1,M)
          PRINT FORMAT HEAD3
          VECTOR VALUES HEAD3=$35H0    HEADPT.       TAILPT.   VALUE  *$
          THROUGH CORNY, FOR J=M,-1, J .L. 1
CORNY     PRINT FORMAT ONE, HEADPT(J), TAILPT(J), VN(J)
          PRINT FORMAT HEAD4,M, OPTVAL
          VECTOR VALUES HEAD4=$11H0 NO.LINKS=I3,16H, OPTIMUM VALUE=I5*$
          TRANSFER TO  START
          END OF PROGRAM
```

MAD Listing of Subroutine MINPAT.

```
$COMPILE MAD,EXECUTE,DUMP,PRINT OBJECT,PUNCH OBJECT                  MINP.
          EXTERNAL FUNCTION (SP,EP,ICN,FCN,VN,NODES,NLNKS,HEAD,TAIL,OPT
         1VAL,CT)
          ENTRY TO MINPAT.
          DIMENSION I(500),U(500),Q(500),V(500)
          NORMAL MODE IS INTEGER
          THROUGH A1,FOR I=1,1,I .G. NODES
          U(I)=999999999
A1        Q(I)=0
          U(SP)=0
          Q(SP)=1
          THROUGH A3,FOR I=1,1,I .G. NODES
          WHENEVER Q(I) .E. 1
              THROUGH A4, FOR A=1,1, A .G. NLNKS
              WHENEVER ICN(A) .E. I
                  WHENEVER U(I)+VN(A) .L. U(FCN(A))
                      U(FCN(A))=U(I)+VN(A)
                      V(FCN(A))=A
                      I(FCN(A))=I
                      Q(FCN(A))=1
                  END OF CONDITIONAL
              END OF CONDITIONAL
A4            CONTINUE
              Q(I)=0
          END OF CONDITIONAL
A3        CONTINUE
          J=EP
          OPTVAL=U(EP)
          CT=1
A5        HEAD(CT)=I(J)
          TAIL(CT)=FCN(V(J))
          VN(CT)=VN(V(J))
          WHENEVER I(J) .E. SP, FUNCTION RETURN
          J=I(J)
          CT=CT+1
          TRANSFER TO A5
          END OF FUNCTION
```

Example Problem No. 61

Data

```
SAMPLE PROBLEM TO TEST NETWORK BEST PATH ALGORITHM   7/18/61
 2          8
30          9
 2          1          58          1          3          45
 1          2          58          1          5         130
 3          5          85          3          1          45
 3          6          66          3          4          75
 2          4          85          3          2          43
 2          3          43          5          8          58
 5          9          40          5          7          32
 5          6          29          5          3          85
 4          3          75          4          6          29
 4          7          28          7          6          27
 7          5          32          7          9          53
 6          3          66          6          5          29
 9          6          62          9          8          25
 6          9          62          6          7          27
 6          4          29          9          5          40
```

Computer Output

```
SAMPLE PROBLEM TO TEST NETWORK BEST PATH ALGORITHM   7/18/61

      START POINT=  2  END POINT=   8
 (I)        (J)       IJ DIST.      (I)        (J)       IJ DIST.
  2          1          58          1          3          45
  1          2          58          1          5         130
  3          5          85          3          1          45
  3          6          66          3          4          75
  2          4          85          3          2          43
  2          3          43          5          8          58
  5          9          40          5          7          32
  5          6          29          5          3          85
  4          3          75          4          6          29
  4          7          28          7          6          27
  7          5          32          7          9          53
  6          3          66          6          5          29
  9          6          62          9          8          25
  6          9          62          6          7          27
  6          4          29          9          5          40

 HEADPT.      TAILPT.    VALUE
  2            3          43
  3            5          85
  5            8          58

NO.LINKS=  3, OPTIMUM VALUE=  186
```

Comments

This problem assumes that the student is familiar with programming in MAD and also with labeling processes which are typical of the network flow algorithms. Successful completion of the Critical Path Scheduling Problem would be one criterion of the student's ability to complete this problem. The network path problem introduces the concept of two or more possible paths in opposite directions as contrasted to the one directional nature of the Critical Path example. The usefulness of switches to keep track of scanned nodes should therefore be pointed out before the student attempts the Network problem.

After successfully completing this problem, the student should be capable of proceeding to an even more complex network flow problem such as a Critical Path Scheduling problem with cost parameters included.

Example Problem No. 62

ENGINEERING ECONOMY

by

Joseph Pistrang

Department of Civil Engineering
City University of New York

Course: Engineering Planning Credit hours: 3 Level: Junior

Introduction

A Junior level course in Engineering Planning begins with a consideration of engineering economy. One of the problems encountered is to investigate the effect of interest rate on the relative economy of alternatives having different first costs and anticipated lives. In particular, it is useful to find that particular value of interest rate which makes any pair of alternatives equally attractive, in terms of equivalent annual costs.

The governing equations are such that a solution by trial is much less awkward than an explicit solution. Since such a solution (by trial) is both tedious and repetitious, it is a likely place to ask for a computer solution.

Problem Statement

Given a pair of alternatives, say A and B, having first costs PA and PB, and lives NA and NB, find the interest rate, I, at which the equivalent annual costs are equal, and also specify which alternative is more economical at higher and lower interest rates.
Note: The actual calculation of interest formulas is only a very small part of the total program, and the problem could be readily extended to include such things as salvage values, costs other than initial costs, and more than two alternatives. This version is as simple as possible.

Instructor's Solution

1) The equivalent annual costs for alternatives A and B (RA and RB) are computed for zero interest rate, using R = P/N where R is annual cost for first cost P and life N. A Boolean flag (ZERBOO) is set to indicate which cost is the larger: 1 if alternative A, 0 if alternative B.

2) The costs RA and RB are then repeatedly computed at interest rates increasing by 1% each time, until the larger R becomes the smaller. For I not equal to zero, $R = P * I / ((1 + I)^{N-1} + 1)$.

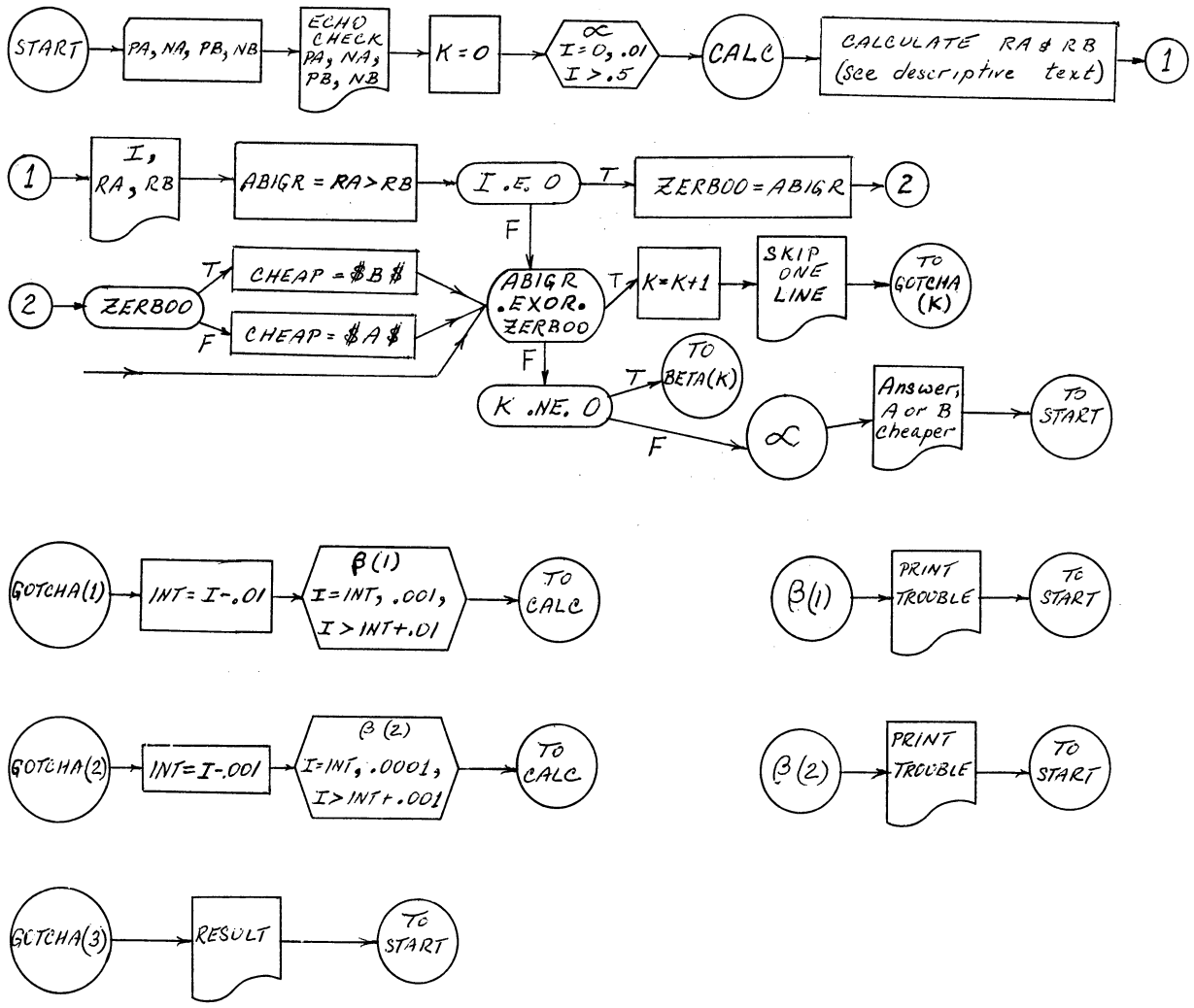
3) I is then reduced to its previous value (reduced by 1%) and the process is repeated, increasing I by 0.1% each time, again until the larger R becomes the smaller.

4) A third go-round, using increments of 0.01% ends the search for the value of I that will make RA = RB.

5) An arbitrary limit of 50% is put on the scope of the first search (step 2); subsequent searches (steps 3 and 4) are automatically limited to ten cycles. If, however, something should go wrong and the subsequent searches are <u>not</u> satisfied in ten cycles, a trouble statement will print a comment.

6) The program was designed to illustrate the use of subscripted statement labels, and the use of Boolean variables, as well as a successive approximation method of solution.

<u>Flow Diagram for Solution</u>



MAD Program and Data

```
 JOSEPH PISTRANG                D042N                 002   005   350
$ COMPILE MAD, EXECUTE, DUMP, PRINT OBJECT , PUNCH OBJECT
          R
          R  TO FIND THE MORE ECONOMICAL OF TWO ALTERNATIVES, A AND B,
          R  AS A FUNCTION OF THE INTEREST RATE, I,  AND DETERMINED BY
          R  FIRST COST IN DOLLARS, P, AND ANTICIPATED LIFE IN YEARS, N
          R
          R                              BY  J PISTRANG
          R
           BOOLEAN ABIGR, ZERBOO
           STATEMENT LABEL GOTCHA, BETA
           DIMENSION GOTCHA(3), BETA(2)
           INTEGER K,NA,NB,C,CHEAP,EXP
START      READ FORMAT DATA,PA,NA,PB,NB
           VECTOR VALUES DATA=$2(F10.0,I2)*$
           PRINT FORMAT DDATA,PA,NA,PB,NB
           VECTOR VALUES DDATA=$20H1  FIRST COST   LIFE/2H0A F9.0,I8/
          1  2H B  F9.0, I8*$
           K=0
           PRINT FORMAT HEAD,$ RATE(I)   ANNUAL COST A   ANNUAL COST B$
           VECTOR VALUES HEAD=$1H0,10C6*$
           THROUGH ALPHA,FOR I=0,.01,I.G.0.5
CALC       WHENEVER I.E.0
           RA=PA/NA
           RB=PB/NB
           OTHERWISE
           RA=PA*(I/((1.+I).P.NA-1.)+I)
           RB=PB*(I/((1.+I).P.NB-1.)+I)
           END OF CONDITIONAL
           PRINT FORMAT COSTS,I,RA,RB
           VECTOR VALUES COSTS=$1H ,F7.4,F17.2,F16.2*$
          R   HERE GOES WITH THE BOOLEANS
           ABIGR=RA .G. RB
           WHENEVER I .E. 0
           ZERBOO=ABIGR
           WHENEVER ZERBOO
           CHEAP=$B$
           EXP=$A$
           OTHERWISE
           CHEAP=$A$
           EXP=$B$
           END OF CONDITIONAL
           END OF CONDITIONAL
           WHENEVER ABIGR .EXOR. ZERBOO
           K=K+1
           PRINT FORMAT SKIP2
           VECTOR VALUES SKIP2 = $1H0*$
           TRANSFER TO GOTCHA(K)
           END OF CONDITIONAL
           WHENEVER K .NE. 0, TRANSFER TO BETA(K)
ALPHA      CONTINUE
           PRINT FORMAT FUNNY,CHEAP,I
           VECTOR VALUES FUNNY=$13H4ALTERNATIVE ,C1,38H IS MORE ECONOMIC
          1AL IF I IS LESS THAN ,F5.4*$
           TRANSFER TO START
```

MAD Program and Data, Continued

```
GOTCHA(1)  INT=I-.01
           THROUGH BETA(1), FOR I=INT, .001, I.G. INT+.01
           TRANSFER TO CALC
BETA(1)    CONTINUE
           PRINT FORMAT TRUBLE
           VECTOR VALUES TRUBLE=$34HOSHOULD NOT HAVE REACHED THIS STEP*$
           TRANSFER TO START
GOTCHA(2)  INT=I- 0.001
           THROUGH BETA(2), FOR I=INT, .0001, I .G. INT + .001
           TRANSFER TO CALC
BETA(2)    CONTINUE
           PRINT FORMAT TRUBLE
           TRANSFER TO START
GOTCHA(3)  PRINT FORMAT RESULT,CHEAP,I,EXP
           VECTOR VALUES RESULT=$13H4ALTERNATIVE ,C1,38H IS MORE ECONOMI
          1CAL IF I IS LESS THAN ,F5.4,12H, OTHERWISE C1,1H.*$
           TRANSFER TO START
           END OF PROGRAM
$ DATA
500.       81000.     20
500.       8900.      20
500.       8800.      20
500.       8700.      20
500.       8600.      20
500.       8500.      20
```

Computer Output for First Two Data Sets

```
  FIRST COST   LIFE
A     500.       8
B    1000.      20

 RATE(I)   ANNUAL COST A   ANNUAL COST B

 0.0000          62.50           50.00
 0.0100          65.35           55.42
 0.0200          68.25           61.16
 0.0300          71.23           67.22
 0.0400          74.26           73.58
 0.0500          77.36           80.24

 0.0400          74.26           73.58
 0.0410          74.57           74.23
 0.0420          74.88           74.89

 0.0410          74.57           74.23
 0.0411          74.60           74.30
 0.0412          74.63           74.37
 0.0413          74.66           74.43
 0.0414          74.69           74.50
 0.0415          74.72           74.56
 0.0416          74.76           74.63
 0.0417          74.79           74.69
 0.0418          74.82           74.76
 0.0419          74.85           74.83
 0.0420          74.88           74.89

ALTERNATIVE B IS MORE ECONOMICAL IF I IS LESS THAN .0420, OTHERWISE A.
```

Computer Output (continued)

	FIRST COST	LIFE
A	500.	8
B	900.	20

RATE(I)	ANNUAL COST A	ANNUAL COST B
0.0000	62.50	45.00
0.0100	65.35	49.87
0.0200	68.25	55.04
0.0300	71.23	60.49
0.0400	74.26	66.22
0.0500	77.36	72.22
0.0600	80.52	78.47
0.0700	83.73	84.95
0.0600	80.52	78.47
0.0610	80.84	79.10
0.0620	81.16	79.74
0.0630	81.48	80.39
0.0640	81.80	81.03
0.0650	82.12	81.68
0.0660	82.44	82.33
0.0670	82.76	82.98
0.0660	82.44	82.33
0.0661	82.47	82.40
0.0662	82.50	82.46
0.0663	82.54	82.53
0.0664	82.57	82.59

ALTERNATIVE B IS MORE ECONOMICAL IF I IS LESS THAN .0664, OTHERWISE A.

Example Problem No. 63

TRAFFIC COUNT ANALYSIS

by

Joseph Pistrang
Department of Civil Engineering
City University of New York

Course: Transportation Credit: 4 Level: Junior

Introduction

To set appropriate highway design standards, it is necessary to estimate the traffic that must be carried in the future. Part of that problem is to estimate present traffic by means of short-time counts of a few hours to a few days (but limited to no more than 24 hours in this problem) which can be, by suitable expansion factors, converted to "annual-average daily traffic" figures (AADT). These expansion factors are obtained from traffic patterns that have been observed elsewhere over a long period, but which are deemed fairly representative of the variation pattern of the road under study. The next step is to convert the present AADT into future estimates; but this will not be considered here.

This problem is essentially one of data processing and reduction, and is one of the first assigned problems in the course.

Problem Statement

Given:

1) Hourly traffic over a full day (typical) H(1)...H(24)
2) Daily traffic over a week (typical) D(1)...D(7)
3) Monthly traffic over a year (typical) M(1)...M(12)
4) A series of short-time-sample traffic counts, called COUNT, taken between the hours T1 and T2 of a particular day called DAY, and month, MONTH.

Write a MAD program which will calculate dimensionless tables and graphs for hourly, daily, and monthly variations, from which expansion factors may be obtained; expand each sample count into an AADT, and obtain their average.

Instructor's Solution

1) The typical hourly figures are converted into a table of cumulative percent of daily traffic, hour by hour; the difference between any two figures, say the 3 p. m. and 9 p. m. percentages, gives the percent of a full day's traffic that can be expected to flow during that interval (say 40%). An actual count of 1000 vehicles during that period would therefore indicate a full day's traffic of 1000/.40 = 2500.

2) The typical daily figures are converted to a table of percent of weekly-average daily traffic:

$$WADT = \sum_{i=1}^{7} [D_i] \ /7 \quad ; \quad \text{and } (\%\ WADT)_i = (D_i/WADT) * 1000.$$

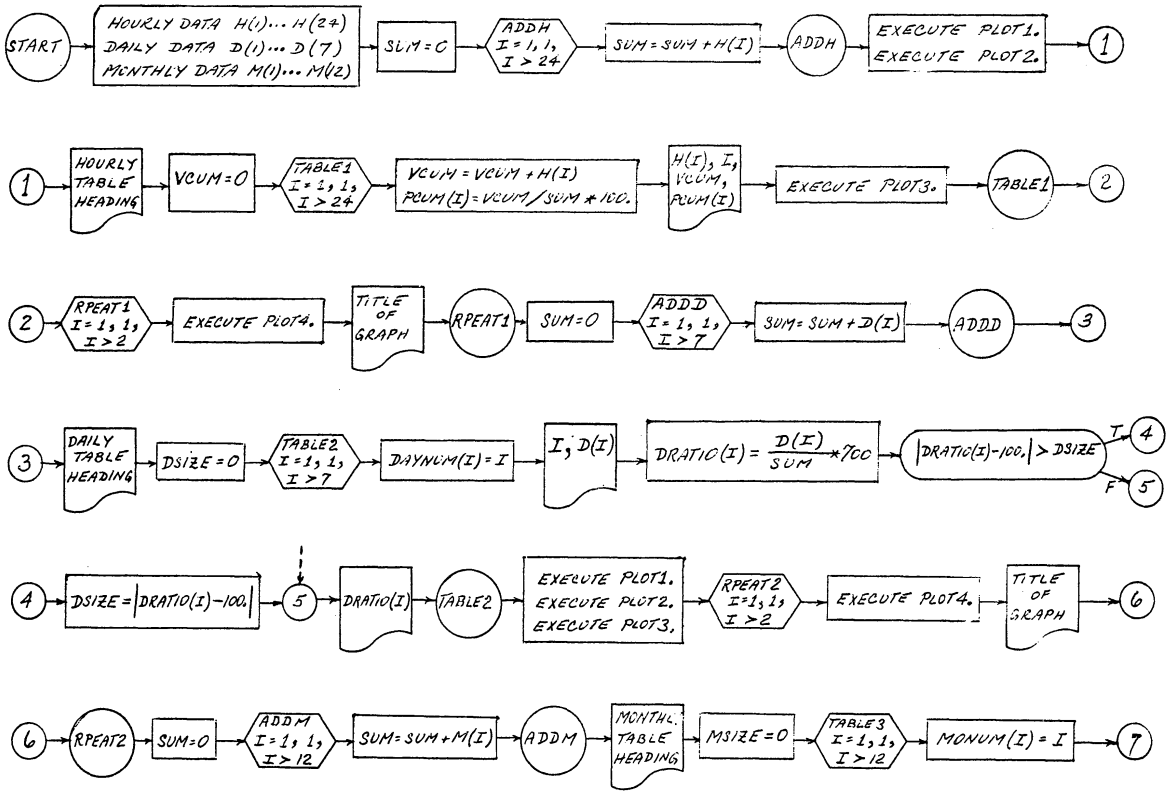
If the previous % WADT were for a Tuesday, for which % WADT is, say, 125%, then the adjusted weekly-average daily traffic for our sample would be 2500/1.25 = 2000.

3) An entirely analogous conversion is made for monthly variations.

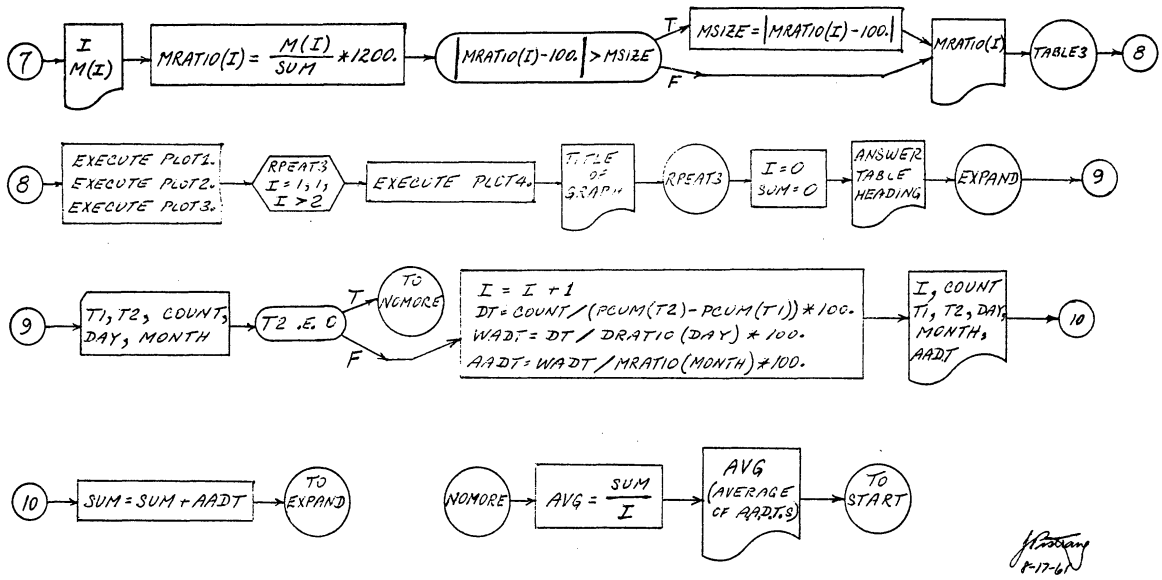
4) The dimensionless tables were plotted using the Carnahan-Evans PLOT subroutine. Maximum and minimum ordinates for the hourly table are known to be 100 and 0. For the daily and monthly graphs, however, the program determines the extreme values and adjusts the ordinate scale of the plot accordingly. Repeat (RPEAT1 etc.) loops were used to produce multiple copies of the graphs.

5) Final output consists of an AADT from each sample, and their arithmetic mean.

Flow Diagram for the Solution



Flow Diagram (continued)



MAD Program

```
 JOSEPH  PISTRANG              D042N              002   015   250
$COMPILE MAD, EXECUTE, PRINT OBJECT, DUMP ,PUNCH OBJECT
          R
          RGIVEN TYPICAL TRAFFIC VARIATION BY HOUR, DAY, AND MONTH, AND
          RGIVEN A SERIES OF SHORT-TIME TRAFFIC COUNTS, THIS PROGRAM
          REXPANDS EACH COUNT INTO AN ANNUAL AVERAGE DAILY TRAFFIC COUNT
          RAND ALSO GIVES THEIR AVERAGE.
          RIT ALSO PRODUCES TABLES AND GRAPHS OF THE TYPICAL VARIATIONS
          R
          R                              BY  J PISTRANG
          R
           DIMENSION H(24),D(7),M(12),DAYNUM(7),MONUM(12),DRATIO(7),
          1 MRATIO(12),PCUM(24),HIMAGE(867),DIMAGE(126),MIMAGE(210)
           EQUIVALENCE (D,DRATIO),(M,MRATIO),(HIMAGE,DIMAGE,MIMAGE)
           INTEGER I, T1, T2, DAY, MONTH
START      READ FORMAT DATA, H(1)...H(24)
           READ FORMAT DATA, D(1)...D(7)
           READ FORMAT DATA, M(1)...M(12)
           VECTOR VALUES DATA=$12F6.0*$
          R
          RPREPARE DIMENSIONLESS TABLE AND PLOT OF HOURLY VARIATIONS
          R
           SUM=0
           THROUGH ADDH, FOR I=1,1, I.G. 24
ADDH       SUM=SUM+H(I)
           PRINT FORMAT HEAD,$       HOURLY  CUMULATIVE  CUMULATIVE$
           VECTOR VALUES HEAD=$1H1,19C6*$
           PRINT FORMAT SUBHD,$ HOUR VOLUME     VOLUME      PER CENT$
           PRINT FORMAT SUBHD,$   0                 0.          0.00$
           VECTOR VALUES SUBHD=$1H ,19C6*$
           EXECUTE PLOT1.(HSCALE,10,5,24,4)
           VECTOR VALUES HSCALE=1,0,-1,0,-1
           EXECUTE PLOT2.(HIMAGE,24.,0.,100.,0.)
```

MAD Program (continued)

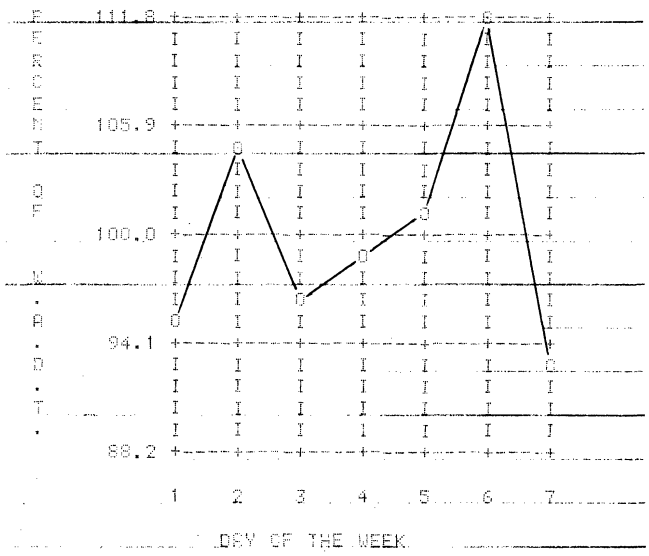
```
            VCUM=0
            THROUGH TABLE1, FOR I=1,1, I .G. 24
            VCUM = VCUM + H(I)
            PCUM(I) = VCUM/SUM*100.
            PRINT FORMAT LINE, H(I), I, VCUM, PCUM(I)
            VECTOR VALUES LINE=$1H ,F12.0/1H ,I4, F18.0, F12.2*$
            IFLO=I
TABLE1      EXECUTE PLOT3.($0$, IFLO, PCUM(I), 1)
            THROUGH RPEAT1, FOR I=1,1, I.G.2
            PRINT FORMAT PAGE
            EXECUTE PLOT4.(44, HMARG)
            VECTOR VALUES HMARG=$     CUMULATIVE PERCENT  OF  DAILY  TRAF
           1FIC $
RPEAT1      PRINT FORMAT HTITLE,$HOUR  OF  DAY$
            VECTOR VALUES HTITLE=$1H0, S50, 3C6 *$
           R
           RPREPARE DIMENSIONLESS TABLE AND PLOT OF DAILY VARIATIONS
           R
            SUM=0
            THROUGH ADDD, FOR I=1,1, I .G. 7
ADDD        SUM=SUM+D(I)
            PRINT FORMAT HEAD,$ DAY   VOLUME   PERCENT OF W.A.D.T.$
            DSIZE=0.
            THROUGH TABLE2, FOR I=1,1, I.G. 7
           R
           RTHE VECTOR DAYNUM IS PREPARED SO THAT ALL DATA POINTS MAY BE
           R PLOTTED BY A SINGLE EXECUTION OF PLOT3.
           R
            DAYNUM(I) = I
            PRINT FORMAT ROW1, I, D(I)
            VECTOR VALUES ROW1=$1H ,I3, F10.0*$
            DRATIO(I) = D(I)/SUM*700.
            WHENEVER.ABS.(DRATIO(I)-100.).G.DSIZE,DSIZE=
           1 .ABS.(DRATIO(I)-100.)
TABLE2      PRINT FORMAT ROW2,DRATIO(I)
            VECTOR VALUES ROW2=$1H+,S20,F8.2*$
            EXECUTE PLOT1.(DSCALE,4,5,6,5)
            VECTOR VALUES DSCALE=1,0,1,0,-1
            EXECUTE PLOT2.(DIMAGE,7.,1.,100.+DSIZE,100.-DSIZE)
            EXECUTE PLOT3.($0$,DAYNUM(1),DRATIO(1),7)
            THROUGH RPEAT2,FOR I=1,1,I.G.2
            PRINT FORMAT HALF
            VECTOR VALUES HALF=$1H2*$
            EXECUTE PLOT4.(20,DMARG)
            VECTOR VALUES DMARG=$PERCENT OF  W.A.D.T.$
RPEAT2      PRINT FORMAT DTITLE,$DAY OF THE WEEK$
            VECTOR VALUES DTITLE=$1H0,S15,3C6*$
           R
           RPREPARE DIMENSIONLESS TABLE AND PLOT OF MONTHLY VARIATIONS
           R
            SUM=0
            THROUGH ADDM,FOR I=1,1, I.G. 12
ADDM        SUM=SUM + M(I)
            PRINT FORMAT HEAD,$ MONTH   VOLUME   PERCENT OF A.A.D.T.$
            MSIZE=0.
            THROUGH TABLE3, FOR I=1,1, I.G. 12
           R
           RPREPARE VECTOR MONUM TO PERMIT ALL POINTS TO BE PLOTTED BY A
           R SINGLE EXECUTION OF PLOT3.
           R
            MONUM(I)=I
            PRINT FORMAT ROW1,I,M(I)
            MRATIO(I)=M(I)/SUM*1200.
            WHENEVER .ABS.(MRATIO(I)-100.).G.MSIZE, MSIZE=
           1 .ABS.(MRATIO(I)-100.)
TABLE3      PRINT FORMAT ROW2, MRATIO(I)
            EXECUTE PLOT1.(MSCALE,4,5,11,5)
            VECTOR VALUES MSCALE=1,0,1,0,-1
            EXECUTE PLOT2.(MIMAGE,12.,1.,100.+MSIZE,100.-MSIZE)
            EXECUTE PLOT3.($0$,MONUM(1),MRATIO(1),12)
```

MAD Program (continued)

```
            THROUGH RPEAT3, FOR I=1,1, I.G.2
            PRINT FORMAT HALF
            EXECUTE PLOT4.(20,MMARG)
            VECTOR VALUES MMARG=$PERCENT OF  A.A.D.T.$
RPEAT3      PRINT FORMAT MTITLE,$MONTH OF THE YEAR$
            VECTOR VALUES MTITLE=$1H0,S20,3C6*$
            PRINT FORMAT PAGE
            VECTOR VALUES PAGE=$1H1*$
           R
           R WE NOW EXPAND THE SHORT-TIME COUNTS
           R
            I=0
            SUM=0
            PRINT FORMAT LIST,$SAMPLE  VEHICLES  PERIOD  DAY  MONTH  A.A.
           1D.T.$
            VECTOR VALUES LIST=$S1,8C6*$
EXPAND      READ FORMAT SAMPLE,T1,T2,COUNT,DAY,MONTH
            WHENEVER T2.E.0,TRANSFER TO NOMORE
            VECTOR VALUES SAMPLE=$2I10,F10.0, 2I10*$
            I=I + 1
            DT=COUNT/(PCUM(T2)-PCUM(T1))*100.
            WADT=DT/DRATIO(DAY)*100.
            AADT=WADT/MRATIO(MONTH)*100.
            PRINT FORMAT EACH,I,COUNT,T1,$-$,T2,DAY,MONTH, AADT
            VECTOR VALUES EACH=$1H0,I4,F11.0,I5,C1,I2,I5,I6,F11.0*$
            SUM=SUM + AADT
            TRANSFER TO EXPAND
NOMORE      AVG=SUM/I
            PRINT FORMAT FINAL,$ AVERAGE A.A.D.T., FROM $,I,
           1$ SAMPLES, IS$, AVG
            VECTOR VALUES FINAL=$1H4, 4C6, I2, 2C6, F8.0 *$
            TRANSFER TO START
            END OF PROGRAM
$DATA
25.   25.   0.    0.    50.   50.   200.  200.  500.  500.  300.  300.
200.  200.  200.  300.  300.  500.  500.  300.  300.  100.  100.  50.
7000. 5000. 4000. 3000. 4000. 6000. 6000.
40000.50000.60000.70000.70000.80000.90000.90000.90000.80000.65000.55000.
         6         10  1000.            3         3
         8         20  4000.            6         6
        10         15  2500.            4        12
        20         24  500.             7        11
0000000000                  000000000000000
725.  350.  200.  100.  100.  225.  1100. 2650. 2150. 1675. 1725. 1700.
1500. 1725. 1950. 2200. 2950. 3300. 2050. 1700. 1300. 1150. 1100. 1100.
26500.29100.26800.27600.28300.31200.25900.
66500.66500.67900.72250.70450.70750.69100.72000.69700.70500.69350.66500.
         6         20   1658.           2         1
        16         21   601.            5         2
        14         24   1033.           7         3
0000000000                 00000000000000000
```

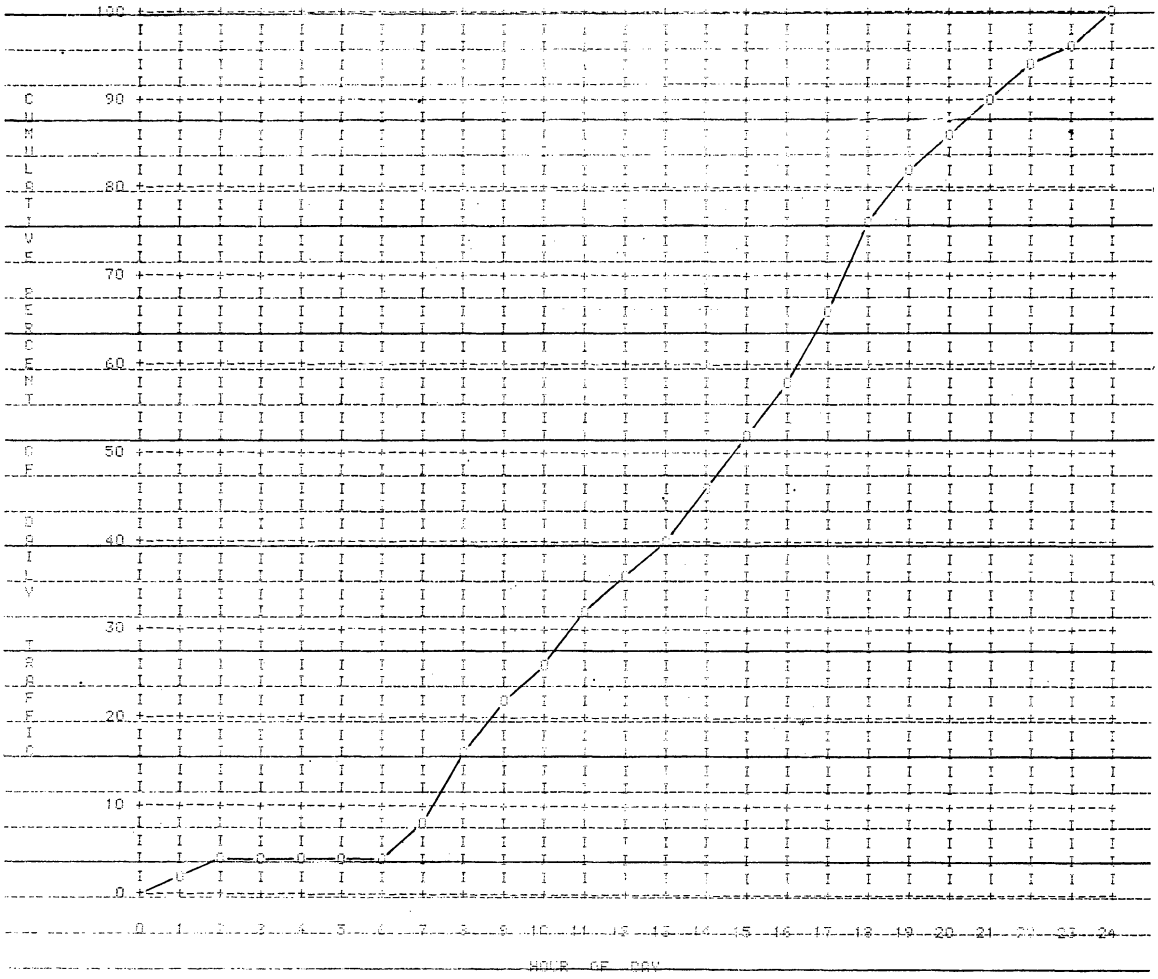
Computer Output (rearranged)

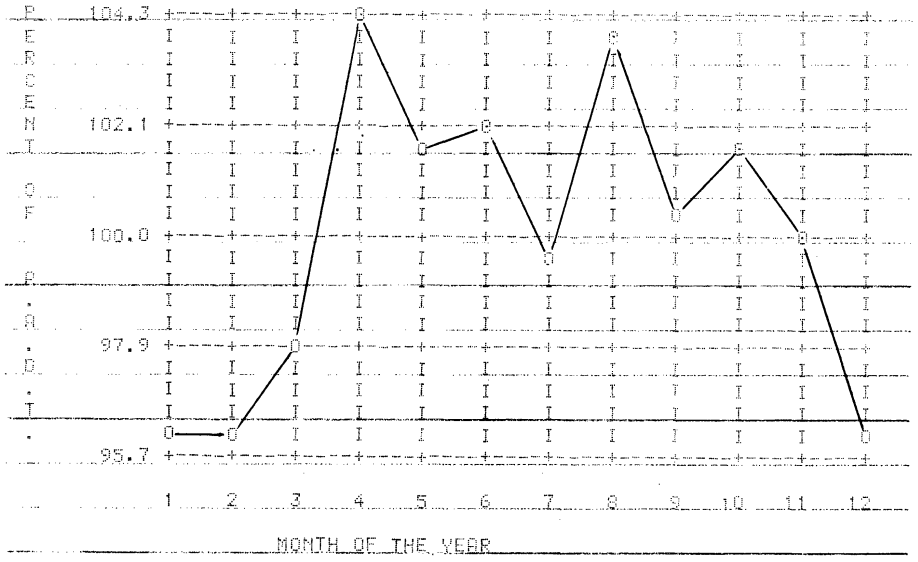
HOUR	HOURLY VOLUME	CUMULATIVE VOLUME	CUMULATIVE PER CENT
0		0.	0.00
	725.		
1		725.	2.09
	350.		
2		1075.	3.10
	200.		
3		1275.	3.67
	100.		
4		1375.	3.96
	100.		
5		1475.	4.25
	225.		
6		1700.	4.90
	1100.		
7		2800.	8.06
	2650.		
8		5450.	15.69
	2150.		
9		7600.	21.89
	1675.		
10		9275.	26.71
	1725.		
11		11000.	31.68
	1700.		
12		12700.	36.57
	1500.		
13		14200.	40.89
	1725.		
14		15925.	45.86
	1950.		
15		17875.	51.48
	2200.		
16		20075.	57.81
	2950.		
17		23025.	66.31
	3300.		
18		26325.	75.81
	2050.		
19		28375.	81.71
	1700.		
20		30075.	86.61
	1300.		
21		31375.	90.35
	1150.		
22		32525.	93.66
	1100.		
23		33625.	96.83
	1100.		
24		34725.	100.00



Example Problem No. 63

Computer Output (rearranged)





Computer Output (rearranged)

DAY	VOLUME	PERCENT OF W.A.D.T.
1	26500.	94.93
2	29100.	104.25
3	26800.	96.01
4	27600.	98.87
5	28300.	101.38
6	31200.	111.77
7	25900.	92.78

MONTH	VOLUME	PERCENT OF A.A.D.T.
1	66500.	95.97
2	66500.	95.97
3	67900.	97.99
4	72250.	104.27
5	70450.	101.67
6	70750.	102.10
7	69100.	99.72
8	72000.	103.91
9	69700.	100.59
10	70500.	101.74
11	69350.	100.08
12	66500.	95.97

SAMPLE	VEHICLES	PERIOD	DAY	MONTH	A.A.D.T.
1	1658.	6-20	2	1	2028.
2	601.	16-21	5	2	1898.
3	1033.	14-24	7	3	2099.

AVERAGE A.A.D.T., FROM 3 SAMPLES, IS 2008.

Example Problem No. 64

MACHINE UTILIZATION

by

Donald D. Deming

Department of Management Engineering

Renssalaer Polytechnic Institute

Course: Production Planning and Control Credit hours: 4 Level: Junior

Statement of the Problem

The purpose of this problem is to demonstrate some of the effects that increasing output has on machine utilization, capital utilization, and the effects these factors have on choice of production plans to achieve most efficient use of facilities, manpower, and materials.

The product under consideration is a valve spring washer, part number H-893. It is being produced in what is known as the Parts Department which produces a large variety of miscellaneous small precision parts for automotive and aircraft engine manufacturers. Much of the department's output is in small lots, some of which are produced regularly, some only infrequently.

It is part of the Planning Department's responsibility continuously to examine the company's many products to be sure that the most appropriate production procedures are used. When products increase in sales to appropriate levels and show promise of continuing sales, consideration must be given to setting them up in specialized groupings of facilities so as to be able to take advantage of specialization in manufacturing.

The Parts Department's machine utilization is approximately 40% of the total machine hours available. Sales for H-893 currently range between 20,000 and 40,000 per month depending on the season and it is anticipated that if costs and prices can be reduced, sales may increase to as much as 60,000 for peak months.

Write a MAD program which uses the following routing data for H-893 to compute results which can be used to draw a horizontal bar chart showing the monthly machine requirements for each machine at the various levels of demand, up to 100,000 pieces per month. Construct a curve showing aggregate machine utilization percentages up to 100,000 pieces per month.

Machine Utilization

Routing for H-893:

Machine	Original Cost	Hours/100 pcs.
Acme 6Sp.	$ 16,500	1.74
Hanchett	8,275	.636
1Sp. Drill	650	.777
Ctrless Grd.	6,000	1.59
Polisher	350	1.60
Internal Grd.	8,700	2.52
Polisher	350	1.27

Notes:
1. The department operates two 8 hour shifts, 5 days per week.
2. A sufficient supply of equipment and space is available.
3. Machine hourly rates are not known but can be related to original cost.
4. Use cross section paper for charts.

The solution for this problem is not shown.

BIBLIOGRAPHY

1. Proceedings of the I.R.E., Vol. 49, #1, Jan., 1961, Computer Issue.

2. Leubbert, W. F., "Candid Comments on Computer Education," Journal of Engineering Education, Vol. 51, #2, Nov., 1960.

3. Gravell, J. P. J., "The Learning Machine," Electronic Progress, Raytheon Corp., March-April, 1961.

4. Minsky, M., "Steps Toward Artificial Intelligence," Proceedings of the I.R.E., Vol. 49, #1, Jan., 1961.

5. McKeachie, W. J., "Understanding the Learning Process," Journal of Engineering Education, Vol. 51, #5, Feb., 1961.

6. Hatch, W. R. and Ann Bennet, "Effectiveness in Teaching," New Dimensions in Higher Education, No. 2, Div. of Higher Education, U. S. Office of Education, 1960.

7. Hatch, W. R. and Ann Bennet, "Research in Teaching Engineering and Related Subjects," Journal of Engineering Education, Vol. 51, #1, Oct., 1960.

8. Balabanian, N., "Thoughts on Engineering Education," I.R.E. Transactions on Education, March, 1961.

9. Russel, J. P., "Teaching of Modern Mathematics to Engineers," Journal of Engineering Education, Vol. 50, p. 437, Feb., 1960.

10. Polya, G., How to Solve It. Princeton University Press, 1954.

11. Malcolm, D. G., "Operations Research and Systems Engineering: The Impact on Engineering Education," Recent Advances in Engineering Sciences, McGraw-Hill Book Co., 1957.

12. Project on "Use of Computers in Engineering Education," Second Annual Report, The University of Michigan, College of Engineering, Ann Arbor, Mich., Dec., 1961.

13. Arden, B., Introduction to Computing Techniques, pre-publication text to be distributed in limited quantity by the Ford Foundation Project during the spring semester, 1962. Later publication by Addison-Wesley Publishing Co. is planned.

14. Organick, E. I., A Computer Primer for the MAD Language, Cushing-Malloy, Inc., Ann Arbor, Mich., 1961.

15. Project on "Use of Computers in Engineering Education," First Annual Report, The University of Michigan, College of Engineering, Ann Arbor, Mich., Aug., 1960.