ac: G

GEORGIA INSTITUTE OF TECHNOLOGY
OFFICE OF CONTRACT ADMINISTRATION
SPONSORED PROJECT INITIATION

Date: *November 28, 1978*

Project Title: *Interactive Heuristics for Large Scale Transportation Models*

Project No: *E-24-682* Green Card

Co-Project Directors: *Dr. J. J. Jarvis and Dr. H. D. Ratliff*

Sponsor: *U. S. Department of Transportation; Transportation Systems Center*

Agreement Period: From *9/27/78* Until 12/31/79 ~~9/26/79~~ *(Performance Period)*

Type Agreement: *Contract No. DOT-TSC-1618*

Amount: *$76,377 (Partially funded at $60,000 through 6/26/79)*

Reports Required: *Master Program Schedule; Master Cost Schedule; Selection and Description Documentation; Monthly Progress Reports; Monthly Cost Reports; Presentations; Demonstrations; Interim Technical Report; Development Documentation (Users Manual, Operations Manual, etc.); Computer Tapes; Final Tech. Rpt.*

Sponsor Contact Person (s):

Technical Matters

Mr. Theodore S. Glickman
Code DTS-223
U.S. Dept. of Transportation
Transportation Systems Center
Kendall Square
Cambridge, MA 02142

(617) 494-2465

Contractual Matters
(thru OCA)

Mr. Robert N. Nelson
Code DTS-852
U. S. Dept. of Transportation
Transportation Systems Center
Kendall Square
Cambridge, MA 02142

(617) 494-2032

Defense Priority Rating: *n/a*

Assigned to: *Industrial & Systems Engineering* (School/Laboratory)

COPIES TO:

- Project Director
- Division Chief (EES)
- School/Laboratory Director
- Dean/Director—EES
- Accounting Office
- Procurement Office
- Security Coordinator (OCA)
- ✓ Reports Coordinator (OCA)
- Library, Technical Reports Section
- EES Information Office
- EES Reports & Procedures
- Project File (OCA)
- Project Code (GTRI)
- Other ______

CA-3 (3/76)

GEORGIA INSTITUTE OF TECHNOLOGY
OFFICE OF CONTRACT ADMINISTRATION

SPONSORED PROJECT TERMINATION

Date: 11/17/80

Project Title: Interactive Heuristics for Large Scale Transportation Models

Project No: E-24-682

Co-Project Directors: Drs. J.J. Jarvis & H.D. Ratliff

Sponsor: U.S. Department of Transportation

Effective Termination Date: 6/30/80

Clearance of Accounting Charges: 9/30/80 (for reporting)

Grant/Contract Closeout Actions Remaining:

- _ Final Invoice and Closing Documents
- _ Final Fiscal Report
- X Final Report of Inventions
- _ Govt. Property Inventory & Related Certificate
- _ Classified Material Certificate
- _ Other __________

Assigned to: Industrial & Systems Engineering (School/XXXXXXXX)

COPIES TO:

Project Director
Division Chief (EES)
School/Laboratory Director
Dean/Director—EES
Accounting Office
Procurement Office
Security Coordinator (OCA)
Reports Coordinator (OCA)

Library, Technical Reports Section
EES Information Office
Project File (OCA)
Project Code (GTRI)
Other OCA Research Prop. Coord.
Project Code (OCA)

CA-4 (1/79)

E-24-682

DOT-TSC-T618 Progress Report

J. J. Jarvis and H. D. Ratliff

October 1%

Since the abstract was not finalized until after the start of classes, other committments prevented Dr. Jarvis from beginning full participation in the project until January 1979. Airline crew scheduling problems as well as a number of local and through freight train scheduling problems were considered as possible demonstration projects for the interactive approach to scheduling being researched. We made several visits with Southern Railway as well as preliminary contacts with Delta Airlines. We determined that both the airline crew scheduling problem, and the train work scheduling problem would be satisfactory problems to demonstrate feasibility of our approach.

Because of the good working relationship we had established with Southern Railway, we decided that the train work scheduling problem was our first choice as a demonstration project. We began considering the various modeling constraints unique to this problem. We also began investigating the various graphic display devices available for implementation of our interactive system once it is developed. We determined that the Chromatics systems would suit our needs very well.

E-24-682

DOT-TSC-T618 PROGRESS REPORT

J. J. Jarvis and H. D. Ratliff

November 19

At our meeting at TSC it was decided that we would delay for the present our selection of a demonstration project. Our efforts would instead be directed toward development of the optimization methodology. The methodology required is a column generation scheme for the underlying set covering problem and a good heuristic for solving the resulting set covering problem.

Since a row pricing scheme is central to both the column generation and the heuristic, we have concentrated our effort on a study of the various pricing mechanisms which we might use. We have determined that a variation of the involutory basic pricing mechanism is superior in several respects to the original method. There is still a lot of work to be done related to what pricing mechanism is best. However, we feel that we have developed considerable insight into possible pricing structure.

E-24-682

DOT-TSC-T618 Progress Report

J. J. Jarvis and H. D. Ratliff

December 76

We continued our exploration of pricing mechanisms. We have proved a theorm which gives necessary conditions for the prices to satisfy in order that we will not exclude in optimum solution as a result of the column generation. We are now trying to determine what form these prices should take in order to be of value to a human interactor.

At the request of Dr. Ted Glickman, we began investigating Dial-A-Ride as a possible demonstration project. Preliminary examination indicates that the static version of this problem may be amenable to the kind of approach that we are studying. At this time, we do not see how to modify our approach to include in a reasonable fashion the dynamic aspects of this problem. We are continuing to study the Dial-A-Ride scheduling problem.

E-24-682

DOT-TSC-T618 Progress Report

J. J. Jarvis and H. D. Ratliff

January 81

Dr. Jarvis will be 50% time for the winter quarter, Dr. Ratliff will remain 25% time. Two Ph.D students are working 1/3 time and one undergraduate student is working 1/3 time. In addition, another Ph.D student is affiliating with the project even though he is not being paid. He is currently on a presidential fellowship but would like to work on the project starting next fall, if it continues.

Georgia Tech has agreed to purchase a chromatics color graphics system (approximate cost - $26,000) for use on this and other contracts. Several graduate assistants began orientation on utilizing the system to display and process the networks. The system is expected to be available sometime in May.

The entire team held several meetings on the design of the overall procedure. One subgroup is focusing on heuristics for improving the solution times for the covering problem component; another subgroup is concentrating on the display mechanisms and formats which would facilitate effective column generation (new candidate routes by the human component). The groups are currently testing and evaluating several alternatives for both parts of the system. All members of the team are also in the process of reading several dozen papers, from the literature, on various aspects of the problem.

The team concentrated on the Dial-A-Ride application. Dr. Ted Glickman arranged a meeting with John Wilson of the Atlanta Regional Commission (ARC) on the Dial-A-Ride systems in the Atlanta area. A meeting was held with Mr. Wilson as which time he gave an overview of the current systems. He also identified a candidate application and is arranging for the Tech team to meet with the operating agency to obtain specific data.

E-24-682

DOT-TSC-1618 PROGRESS REPORT

J. J. Jarvis and H. D. Ratliff

February

Drs. Jarvis and Ratliff met with Fred Coleman of Marietta-Cobb Community Service Center, and separately with Ruth Zaleon of Senior Citizen Services. Each of these agencies provides demand responsive transportation systems.

We discussed at length the currently operational procedures for the two demand responsive systems. We were able to gain considerable insight into dial-a-ride as these people see it. We concluded that some additional methodology is required if these systems are to be run as envisioned. We also concluded that any consolidation of systems will require a better scheduling methodology than the present if the combined system is to function at any reasonable level. The primary reason that those systems work as currently constituted is that they are small.

We have also been working on price directed heuristics for set covering problem. This class of heuristics appears to have great potential.

E-24-682

DOT-TSC-1618 PROGRESS REPORT

J. J. Jarvis and H. D. Ratliff

March

Our emphasis this month has been on the development of models which can aid in the interactive column generation process. We have broken the column generation process into three components: pricing, clustering, and chaining. We have previously developed some heuristics for pricing. This month we developed three fundamental models as candidates to aid in clustering. All three are variants of location/allocation models. We are modifying them to take advantage of human interaction. We have developed an assignment based model to "chain" together the clusters. The chained clusters will then go into the set covering model.

E-24-682

DOT-TSC-1618 PROGRESS REPORT

#20

J. J. Jarvis and H. D. Ratliff

April

Considerable time was spent in early April preparing for the conference on Transportation, Network Analysis and Control at the Transportation System Center. Drs. Jarvis and Ratliff had been invited to present their research at that conference.

Also during April, the Georgia Tech team received its Chromatics Colorgraphics terminal. Two of the graduate research assistants were assigned the duties of learning to operate this system and to integrate it into the overall scheduling system.

Effort continued on refining the pricing mechanisms as well as gaining additional experience with the clustering models. Feedback from individuals at TSC, responsible for dial-a-ride, indicated that the Georgia Tech team were developing promising models and methods for routing and scheduling dial-a-ride vehicles. Some consideration was also given to the larger class of problems solvable by the techniques being developed. Example applications in this class include: the moving van problem, train work scheduling, freight haul scheduling, and certain military logistics applications.

xc: Al Becker

6-13-79

E-24-682

DOT-TSC-1618 PROGRESS REPORT

J. J. Jarvis and H. D. Ratliff

May

During this month, substantial progress was made in developing and testing computerized codes for clustering, and for network generation and human interface with the Chromatics Colorgraphics terminal. Testing of the computer code for clustering uncovered several difficulties with the basic clustering model. Various mathematical approaches to modifying these models to alleviate these difficulties were tried.

As the team gained more experience with the Colorgraphics terminal, the overall interactive philosophy began to take on new direction. In addition, prototype testing of example dial-a-ride problems with fifty trips indicated difficulties with parts of the overall scheduling system. This led to revisions of portions of the interactive philosophy.

Several new results were developed, with necessary proofs, for improving the pricing mechanism associated with the covering model. In addition, the emerging interactive philosophy, now encompasses two covering models: one associated with clustering and the other associated with chaining.

xc: Al Becker

6-13-79

E-24-682

DOT-TSC-1618 PROGRESS REPORT
J. J. Jarvis and H. D. Ratliff

June, 1979 Thru August, 1979

These summer months were spent developing the demonstration software packages, as per contract requirements. The demonstration was conducted on August 24 with TSC and UMTA/NBS Officials present. The software links the Chromatic Colorgraphics computer to the CYBER 74 computer located on the Georgia Tech campus.

The software residing on the CYBER 74 consist of a number of program modules. One program module, the driver, handles all information flows between the Chromatics and the CYBER. This driver module generates and maintains all of the necessary data base for the operation of the dial-a-ride routing and scheduling system. As second program module on the CYBER handles the solution of the associated covering problems generated in the dial-a-ride scheduling process. This same module also develops the necessary prices used in other portions of the algorithm. A third module handles all of the clustering requirements. This includes implementation of the location allocation model as well as implementation of the pricing concept. The clustering module is particularly flexible in that it permits fixed point as well as Euclidean clustering approaches. It also accepts varying parameters in each of these models. A fourth program module residing on the CYBER, handles all of the routing requirements within the various clusters. This module consists of a branch and bound superimposed on a travelling salesman algorithm. The branch and bound is necessary because of the sequencing constraints resulting from the fact a patron must be first picked up before he/she is dropped off. The last major module in the CYBER package consist of a network flow algorithm for the chaining process. This algorithm is variant of the out-of-kilter algorithm for network flows.

The software residing on the Chromatics is designed to handle all of the display capability on the screen, the acceptance of information from the CYBER, and the acceptance of imput from the human, through light pen and/or keyboard. The data base requirements and associated software on the Chromatics side of the total interactive optimization system are much more complex. This is a result of the fact that the human can perform certain functions (e.g. routing) himself or he can request routing from the CYBER. This same flexibility extends to other aspects of the scheduling system including clustering and chaining.

The data base for the August 24 demonstration consisted of 27 randomingly generated trips in a typical dial-a-ride example. During the demonstration it was suggested that the system might be best tested on a set of real data coming out of either Cobb County Georgia, which the Georgia Tech team has visited, or Rochester, New York, which TSC officials had data for.

<u>September 1979 Thru December, 1979</u>

During these fall months the Georgia Tech team concentrated its effort in three distinct areas. Graduate students revisited the Cobb County agency and collected data on several weeks request for dial-a-ride service. Members of the Georgia Tech team are involved in reducing and coding the data obtained. Officials from TSC supplied the Georgia Tech team with a tape of data on the Rochester dial-a-ride system. Others members of the team are involved in stripping the data off this tape and processing it into a form suitable for input to the interactive optimization package. A second major effort of the team has been the continued development and refinement of modules and systems for the interactive optimization package. A linear clustering model has been designed, developed and is in the process of being tested. Rather than clustering to a trip or a pseudo trip, the concept here is projection on to a straight line which would represent essentially the mean locations among the trips. The expectation is that all of the models for clustering will be subjected to a rigorous analysis and testing phase to determine those most appropriate for the optimization package.

Continued development and refinement of the chaining model has progressed along the lines of application of penalty function concepts Such techniques are under consideration to handle the side constraints generated by the requirement that a trip not be in more than one cluster in a given chain. In the penalty function approach these "nasty" constraints are multiplied by an appropriate penalty and combined into the objective function, leaving a "nice" network flow problem remaining for which efficient solution techniques exist. The penalty function procedure proceeds with a search over appropriate values for the penalties.

The third major thrust during the fall months has been a concentration on the development of techniques and modules which are based on more intuitive insights into the dial-a-ride scheduling process. This focus on intuitive algorithm development is necessitated by a desire to develope scheduling packages that can be implemented totally on the Chromatic Colorgraphics system, without requirement for utilization of the CYBER computer. A savings heuristic has been developed for clustering in the dial-a-ride process. This savings approach implements the pricing mechaniam to aid in adding one trip at a time to a cluster being built. Initial testing of the savings approach appears to be very favorable.

E-24-682

ACTUAL & PROJECTED EXPENDITURES ON CONTRACT TSC/22-0029-RN June 13, 1979
(GEORGIA TECH #E-24-682)
J. J. JARVIS & H. D. RATLIFF
(AMOUNT/CUMULATIVE AMOUNT)

Month	(1) Personal Services Principal Investigators	(2) Personal Services Graduate Assistants	(3) Retirement on (1)	(4) Overhead on (1)+(2)	(5) Travel	(6) Computer	(7) Materials & Supplies	(8) Total
October, 1978	$ 698 (43 hrs.)	$ 0	$ 69	$ 530	$ 0	$ 0	$ 0	$ 1,297
	698	0	69	530	0	0	0	1,297
November	698 (43 hrs.)	0	69	530	462	0	0	1,759
	1,396	0	138	1,060	462	0	0	3,056
December	698 (43 hrs.)	0	69	530	0	0	3	1,300
	2,094	0	207	1,590	462	0	3	4,356
January, 1979	1,890 (130 hrs.)	0	186	1,436	0	0	2	3,514
	3,984	0	393	3,026	462	0	5	7,870
February	1,890 (130 hrs.)	0	186	1,436	0	0	0	3,512
	5,874	0	579	4,462	462	0	5	11,382
March	1,890 (130 hrs.)	0	186	1,436	0	0	147	3,659
	7,764	0	765	5,898	462	0	152	15,041
April	1,890 (130 hrs.)	933 (116 hrs.)	186	2,145	567	0	57	5,778
	9,654	933	951	8,043	1,029	0	209	20,819
May	1,294 (87 hrs.)	933 (116 hrs.)	127	1,693	182	0	250	4,479
	$10,948	$1,866	$1,078	$ 9,736	$1,211	$ 0	$459	$25,298
PROJECTED THROUGH CONTRACT DURATION								
June	$ 3,881* (87 hrs.)	$1,185 (151 hrs.)	$ 381	$ 3,851	$ 0	$1,500	$ 37	$10,835
	14,829	3,051	1,459	13,587	1,211	1,500	496	36,133
July	5,402 (260 hrs.)	1,875 (134 hrs.)	568	5,531	0	1,500	0	14,876
	20,231	4,926	2,027	19,118	1,211	3,000	496	51,009
August	5,402 (260 hrs.)	1,875 (134 hrs.)	568	5,531	750	1,500	0	15,626
	25,633	6,801	2,595	24,649	1,961	4,500	496	66,635
September	5,402 (260 hrs.)	0	568	4,106	0	0	300	10,376
	$31,035	$6,801	$3,163	$28,755	$1,961	$4,500	$796	$77,011†

* This amount includes three (3) checks paid to individuals on academic year contracts, since these contracts are paid in twelve (12) checks.

† Some adjustment in individual category amounts will be necessary to bring this figure to within allocated contract amount.

Master Cost Schedule
Monthly cost rpts thru ~~[illegible]~~ May 1979

xc: Al Becker

E-24-682

FINAL REPORT

INTERACTIVE HEURISTICS FOR LARGE SCALE TRANSPORATION MODELS

Submitted By:

Principal Investigators
John J. Jarvis
H. Donald Ratliff

Graduate Research Assistants
Frank H. Cullen
David J. Friedman
Robert T. Lewis

Submitted To:

U.S. DEPARTMENT OF TRANSPORTATION
TRANSPORTATION SYSTEMS CENTER

Under
Contract No. DOT–TSC–1618

August 1980

GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL OF INDUSTRIAL & SYSTEMS ENGINEERING
ATLANTA, GEORGIA 30332

E-24-682

FINAL REPORT

CONTRACT NO. DOT-TSC-1618

INTERACTIVE HEURISTICS FOR LARGE SCALE TRANSPORTATION MODELS

Submitted To:

U. S. Department of Transportation

Transportation Systems Center

Submitted By:

John J. Jarvis

H. Donald Ratliff

Principal Investigators

Frank H. Cullen

David J. Friedman

Robert T. Lewis

Graduate Research Assistants

School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332

August 1980

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

The research team wishes to thank Bob Crosby of the United States Department of Transportation (U.S. D.O.T.) and to Ed Roberts and Ted Glickman of U.S. D.O.T.'s Transportation Systems Center in Cambridge, Massachusetts for their guidance, support and encouragement. They provided the framework and insights into the Dial-A-Ride application area.

The project team also wishes to thank Paul Connolly of the Transportation System Center for providing dial-a-ride data for Rochester, New York. He and his staff were extremely helpful in coding and interpreting the data for the CYBER 74 computer at Georgia Tech.

Finally, the team acknowledges the tremendous help of Fred Coleman and his staff at Marietta - Cobb Community Service Center in Cobb County, Georgia for supplying insights and data on their dial-a-ride system.

ABSTRACT

This report summarizes the results of an intensive study into the application of interactive heuristics in transportation systems design and analysis. Project results include (1) the development of new methodology to aid researchers and practitioners and (2) a set of computer codes which were used to test concepts and to demonstrate the potential usefulness of the approach. Human Aided Optimization/Heuristics will clearly prove more important in the 1980's as we learn more about those functions which humans can perform better than computers (such as spatial pattern processing), and find unique ways to integrate these capabilities into a human/computer system.

In the area of methodology, the project has resulted in the development of new theorems and lemmas detailing the concept of "prices" for set covering and set partitioning problems. The prices are similar to those for linear programs, however, because they have been developed for integer programs they don't enjoy some of the nicer properties of linear programming dual variables. Set partitioning models are developed for the dial-a-ride and delivery problems. The research suggests a decomposition of these complex problems by employing two concepts called "clustering" and "chaining".

Clustering involves the grouping of trips, bus stops, demand points, etc. into logical subsets which should be served by the same vehicles. The report presents several models for clustering, as well as supporting theorems and lemmas. The application of clustering models to United States Department of Transportation data for the Rochester, NY Dial-A-Ride transit system are also presented. Our tests indicate that there is significant potential for this kind of approach to demand responsive transportation routing and scheduling.

The concept of chaining was developed to provide a means for linking the clusters into logical vehicle routes. Several models were developed and tested for chaining. A Lagrangian relaxation approach has been developed and tested on sample data. More research is required before chaining can become a function useful for immediate practical implementation.

To test the methodology and concepts developed, a human aided (heuristic) optimization system was developed for the dial-a-ride and delivery problems. The computer system consisted of a Chromatics Color Graphics (high resolution) computer system for presentation of visual/spatial information to the human and a CDC CYBER 74 computer system to perform the optimization functions in support of the human aided optimization process. For dial-a-ride problems, these codes were tested on sample data sets and on data sets for Rochester, New York and Cobb County, Georgia.

The project also tested the feasibility of a stand alone mini computer interactive heuristics system for transportation systems analysis. Codes, which reside entirely on the Chromatics computer, were developed and tested on a series of delivery problems. Three "classic" delivery test problems - a 50-point, a 75-point, and a 100-point problem - were used. These three problems have been tried, during the last decade, by dozens of researchers with different approaches to (fully automatic) computer systems. Due to their complexity, the optimal solutions to these problems are not known. This report shows that, in the 50 and 75-point problems, after a few iterations the human aided computer system was able to obtain the best known solution. In the case of the 100-point problem the system was able to obtain a better solution than any yet obtained. These results provide significant encouragement for the approach outlined in this report.

I. INTRODUCTION

This report summarizes research into interactive heuristics for large scale transportation systems analysis and design. The research suggests that by employing the human capability for spatial pattern processing together with the immense computing power of today's computers, both the human's and computer's ability to handle complex transportation problems is enhanced.

The bulk of this final report consists of three technical reports which were produced in the three major areas of research on the project. These three areas include: (1) an overall modeling process for interactive heuristics in routing, (2) concepts and models in clustering and (3) concepts and models in chaining. The technical reports detailing each of these areas are contained in Appendix A of this final report.

Certain of the models developed under this contract were coded in FORTRAN for a CDC CYBER 74 and BASIC for a Chromatics CG 1999 color graphics computer. These codes were tested against (1) routing data from the literature, (2) data from Rochester, NY and (3) data from Cobb County, GA. These codes are presented in Appendix B of this report.

II. METHODOLOGY

The fundamental modeling approach taken in this research is to employ a set partitioning formulation of the routing problems studied (dial-a-ride and delivery). A set partitioning matrix has one row for each trip (or point to be visited) and one column for each feasible route. Thus, while the matrix has a reasonable number of rows it has an enormous number of columns.

To circumvent the column difficulty we have suggested a column generation approach which only identifies columns as they appear to be favorable to producing a better solution. In order to accomplish this, quantities called "row prices" have been developed. These prices act like dual variables in linear programming except that they are associated with integer programs (the set partitioning problem).

The details of the modeling methodology are given in technical report #J-80-1 in Appendix A. In the interactive process, the computer solves the set partitioning problem at hand, generates the row prices and suggests candidate trips or delivery points to examine next (based on a savings approach). The human guides the overall process, accepts, rejects or develops candidate routes, and provides the stopping criteria for the algorithm.

Especially important in the dial-a-ride problem, is the need to decompose the complex task of developing good candidate routes. Thus, two decomposition concepts - clustering and chaining - were developed. In clustering we seek to identify those trips whose origins and whose destinations are reasonably close together so that they would logically be picked up and delivered by the same vehicle. These individual trips are then replaced by a single pseudo (cluster) trip. Technical report #J-80-15, in Appendix A, describes several models and procedures for clustering.

Once good cluster trips are identified the focus is then shifted to chaining (linking) these clusters into feasible routes. Models and algorithms for chaining are developed in technical report #J-80-16, in Appendix A.

The dial-a-ride procedure has been coded for a linking of a Chromatics CG 1999 color graphics computer with a CDC CYBER 74 computer. The cyber performs large scale computing functions such as set partitioning solution generation, minimal cost network flow computation, solution of certain clustering models, and application of a traveling salesman algorithm. The Chromatics color graphics computer handles various display functions and simple data manipulation. In addition, a special data development program was coded for Cobb County, GA dial-a-ride data.

A delivery algorithm has also been coded for the same computer combination. The cyber generates traveling salesman solutions as required, and the chromatics performs all other functions. The current operating version does not employ the set partition solution procedure. Instead entire new solutions are generated at each iteration.

III. MICROCOMPUTER STAND ALONE CAPABILITY

A version of the dial-a-ride algorithm which resides entirely on the chromatics color graphics computer, was coded and tested. This version does not use the sophisticated minimal cost network flow algorithmic, traveling salesman procedure in set partitioning algorithm. Instead it utilizes the human to perform these functions.

Moderate testing of the stand alone code indicates that it can perform reasonably well in a dial-a-ride routing and scheduling environment. However, this version does require much greater involvement of the human in solution development.

The delivery code was developed in such a way that, if the human did not request a traveling salesman solution, the chromatics computer could act independently. The only purpose of the traveling salesman procedure is to guanrantee an optimum route among the set of selected delivery points. Since the human usually does a very good job at developing new optimal routes as he/she is building them, this procedure currently acts much better as a stand alone method.

IV. APPLICATION OF THE METHODOLOGY

Each of the computer codes was thoroughly tested on a number of randomly generated test problems. In addition, tests were conducted with data provided from three other sources: the literature, U. S. Department of Transportation officials, and local Atlanta dial-a-ride administrators.

The delivery algorithm was tested on three problems from the literature (see report #J-80-1). These were a 50-city, 75-city and 100-city delivery problem. As indicated in the report, the delivery algorithm performed quite well in every case (equaling or exceeding the best known solution). These results provide much encouragement for further development and testing of the method.

As a first test of the dial-a-ride procedure, several weeks data was taken for a dial-a-ride transit system in Cobb County, GA. The system consists of 4 vehicles and handles approximately 50 daily trips.

A special data input program (see Appendix B) was developed to allow the graphical input of data for this system. Figure 1 gives an example of the input data. The computer generates the area maps and the human operator inputs the trips (straight lines and boxes) by simply pointing a light pen at the origin and destination of each trip.

Data was entered for a typical weekday a.m. operation (8:00 a.m. - 12:00 noon) of the Cobb County dial-a-ride system. While comparison data was unavailable, the computerized dial-a-ride procedure appeared to perform quite well in generating good vehicle routes.

Finally, the models were applied to a sample cluster set for the Rochester, NY dial-a-ride system. Particular emphasis was given to the applicability of the clustering models in a "real-world" environment. The following figures illustrate results of the application. In

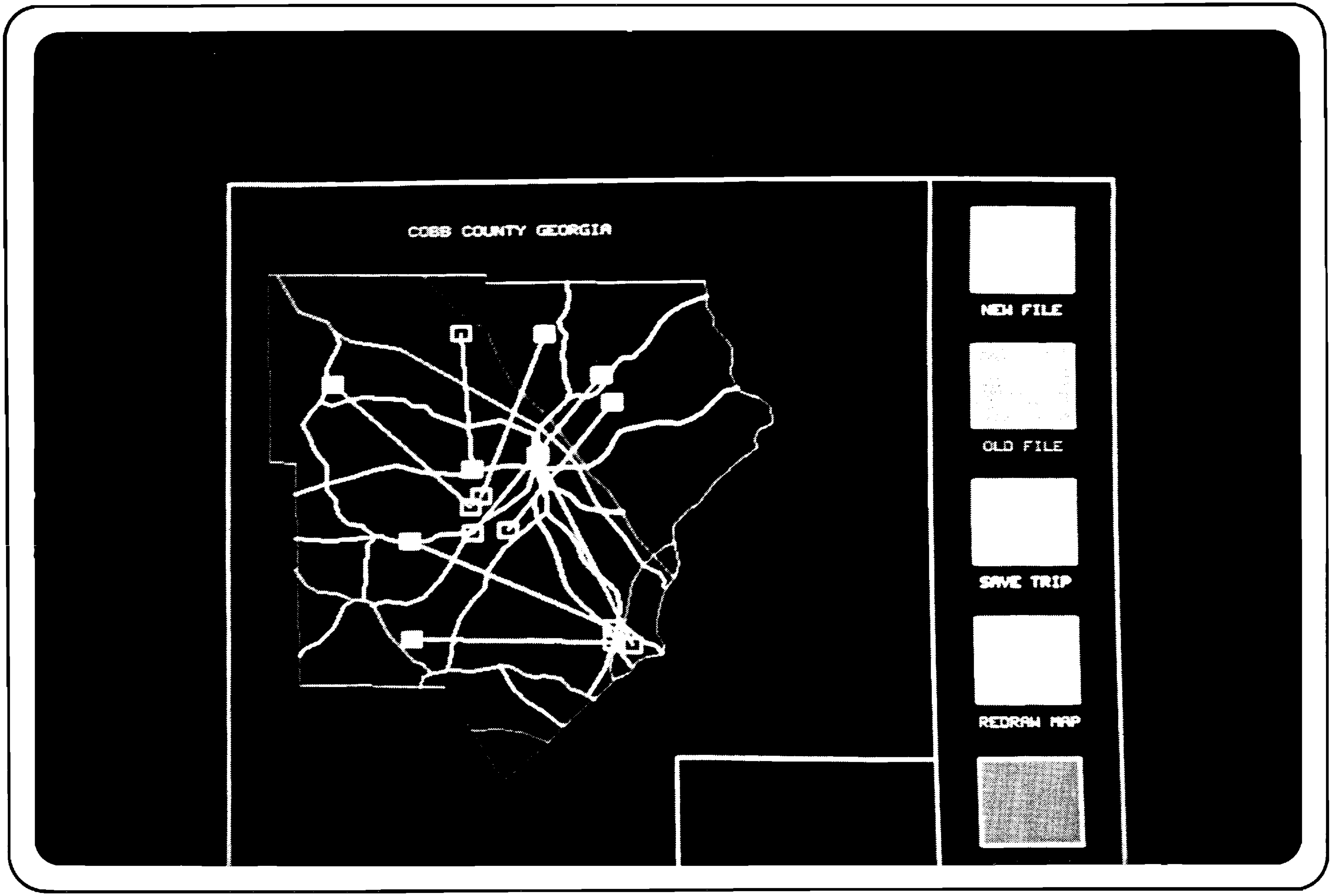


Figure 1. An Example of the Cobb County Data Input Program

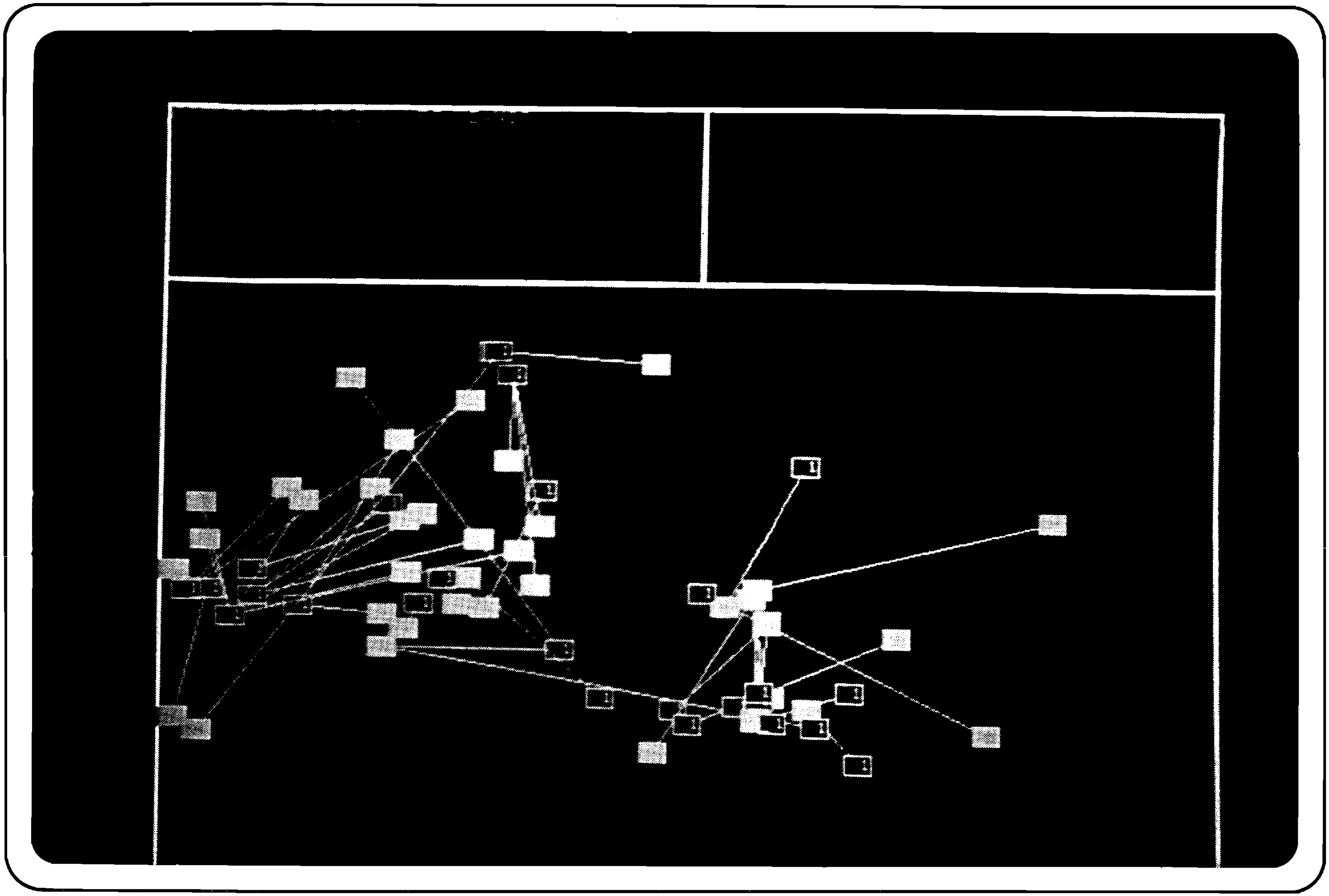


Figure 2. A Sample of 42 Trips for the Rochester Dial-A-Ride System

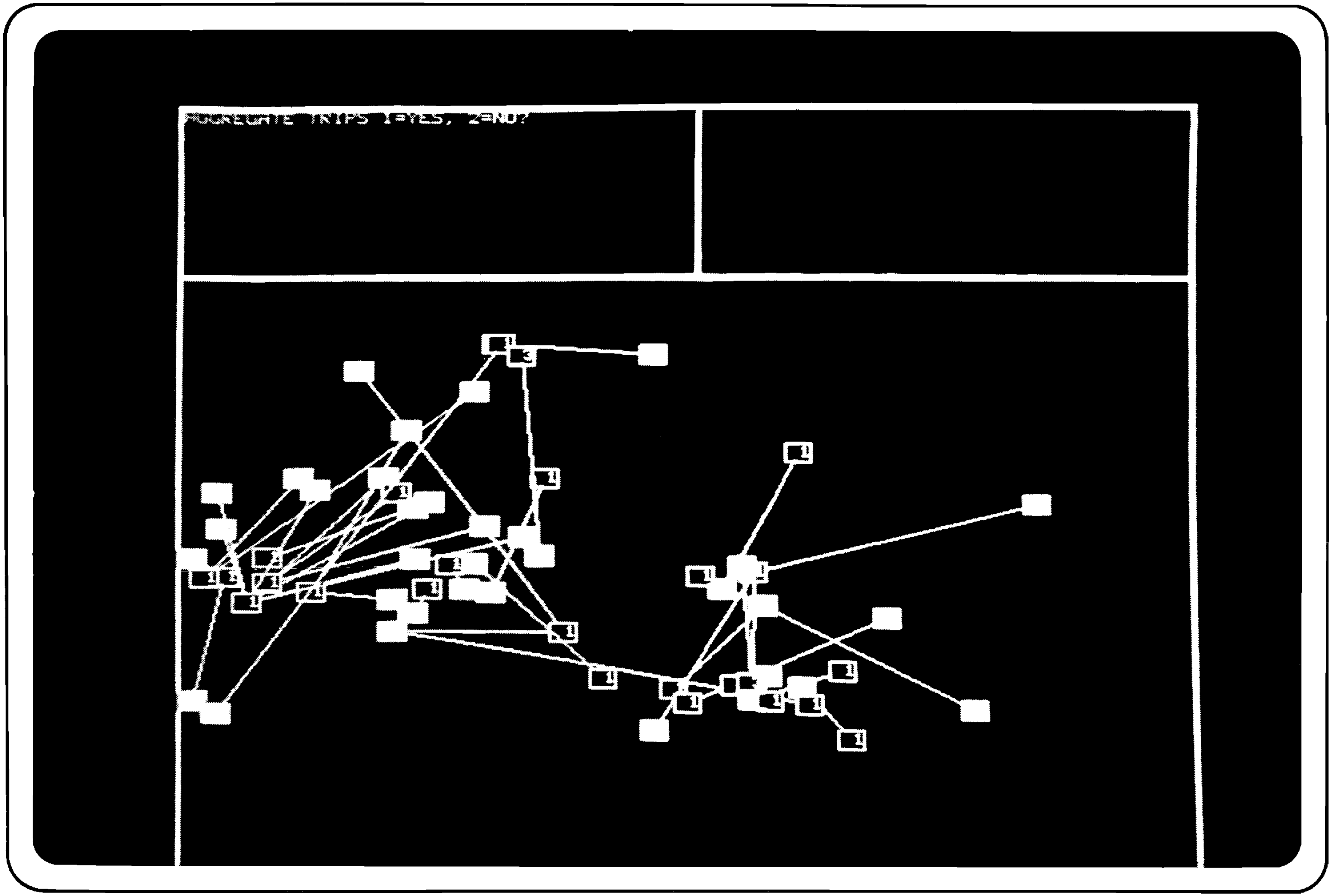


Figure 3. Preprocessed Rochester Dial-A-Ride Data

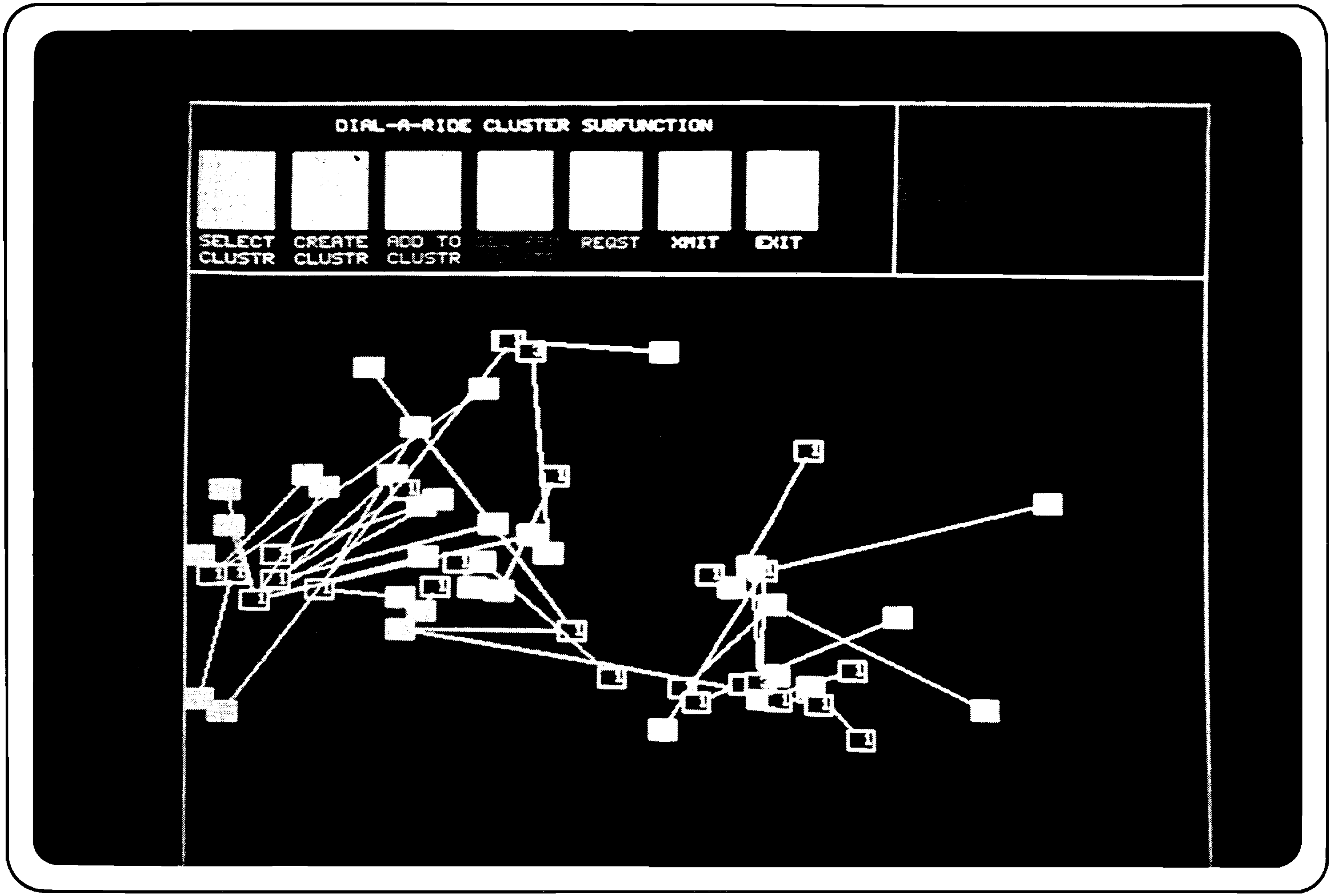


Figure 4. A Request for Six Clusters of Not More Than 10 Trips Each

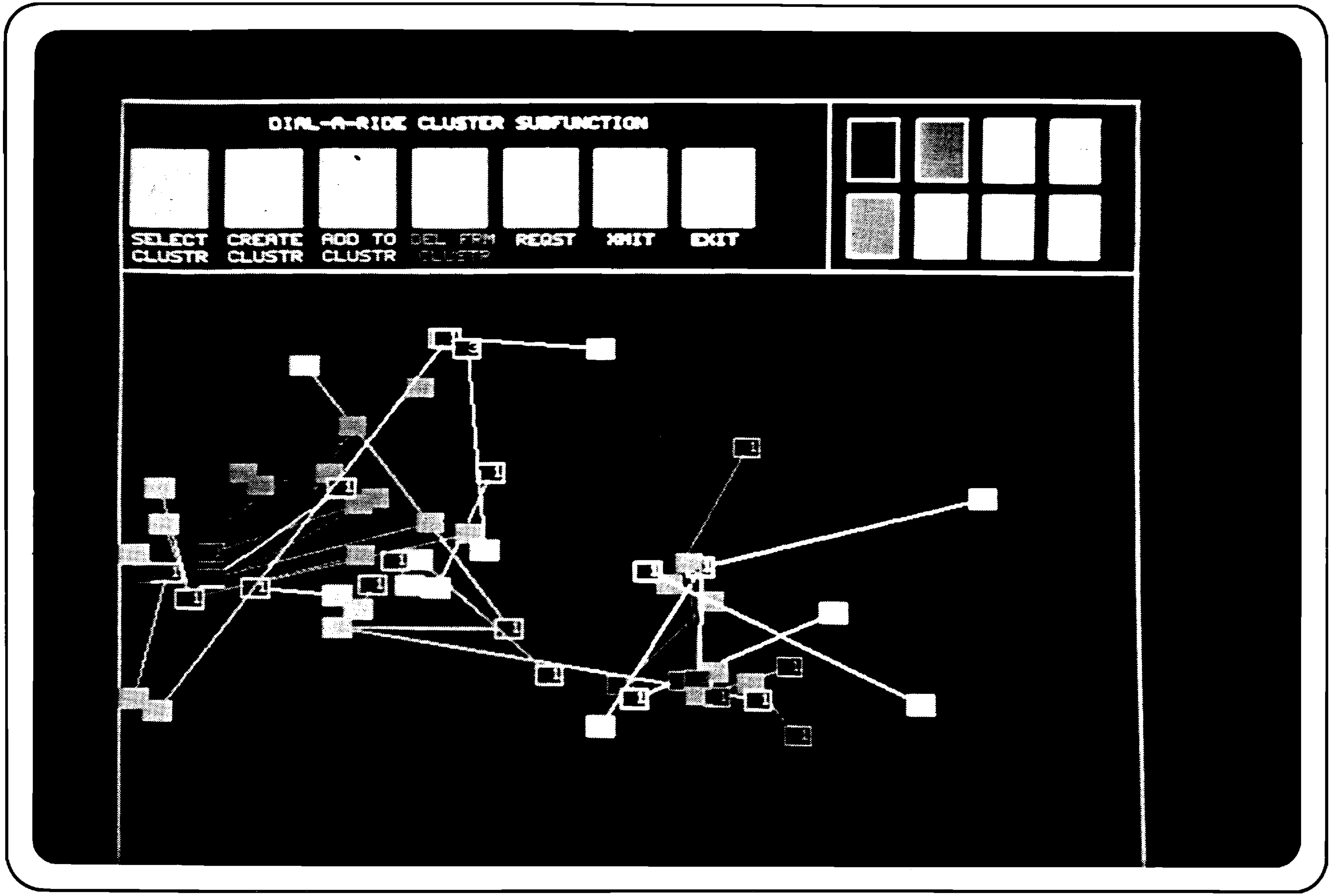


Figure 5. Results of the Cluster Request of Figure 4

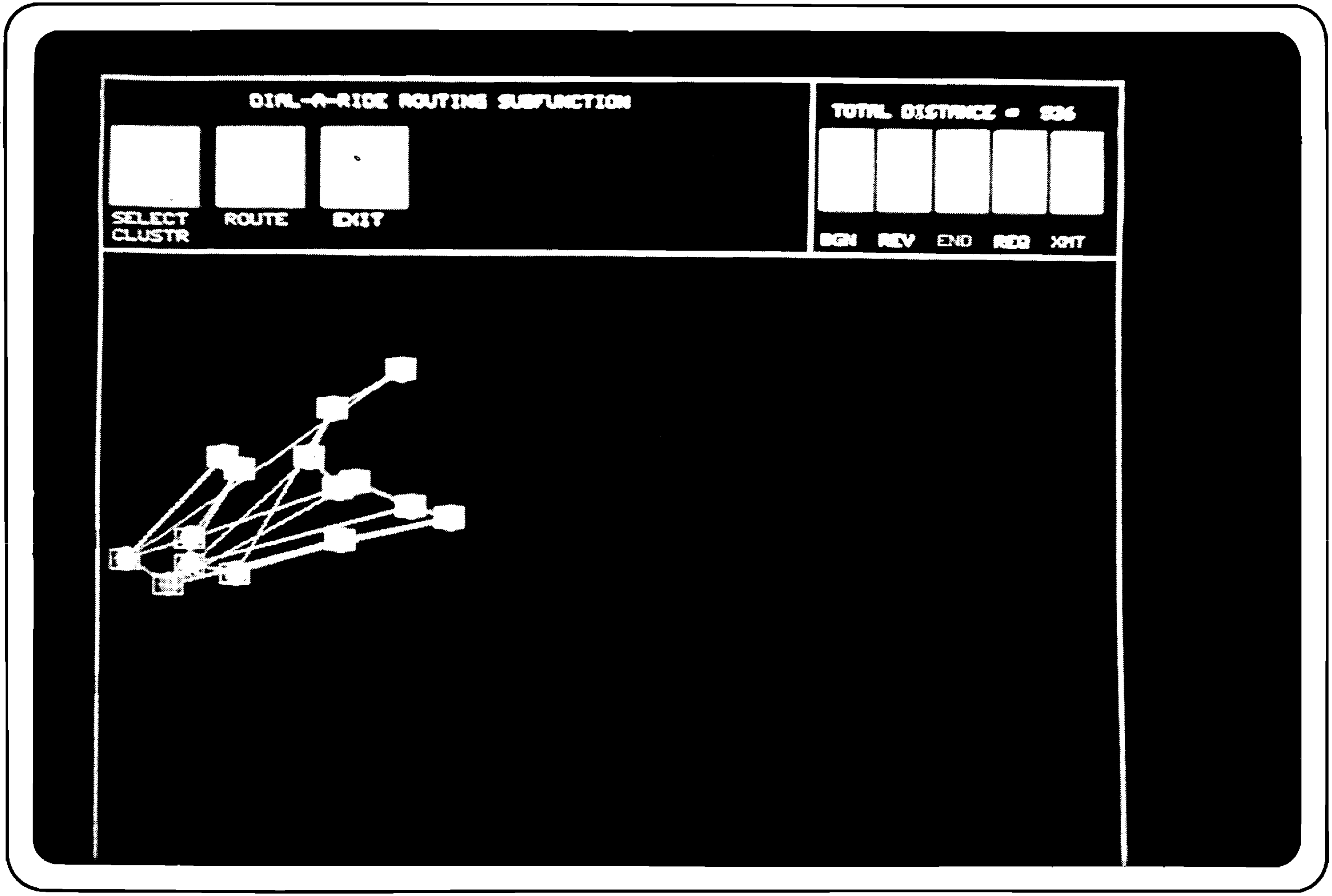


Figure 6. Routing Through A Cluster

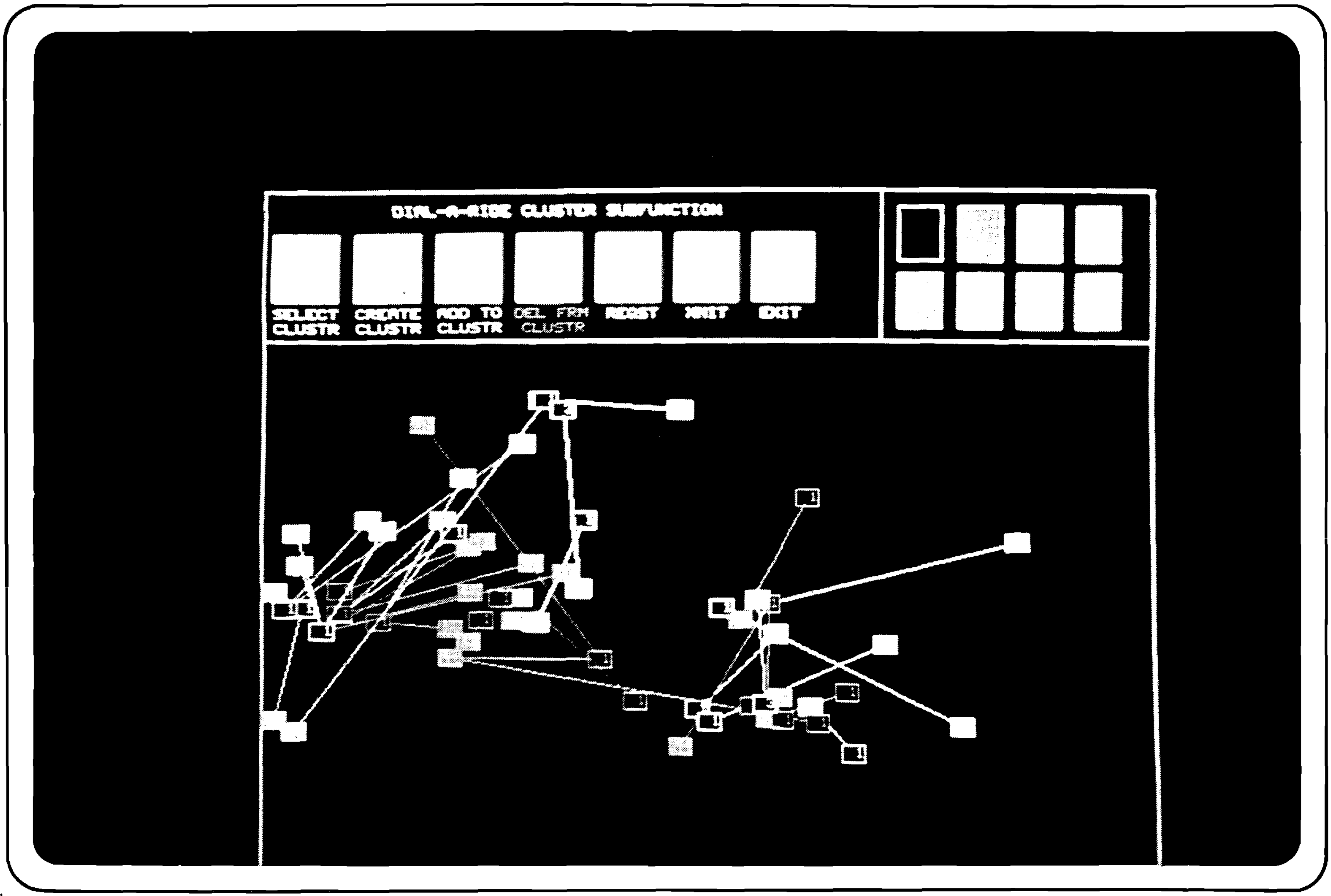


Figure 7. Results of a Ten Cluster Request (Dark Blue, Light Blue, Red and Green Repeated)

general, the models performed quite well. However, this application pointed to a special difficulty which had to be overcome before the application could be completed.

In the Rochester data selected, many of the origins and many of the destinations were the same. While this enhanced the capability of the clustering algorithms, the human was subjected to problems of clutter due to the closeness of different trips. A special preprocessing program was developed which combined several trips into a single cluster trip for further processing. After this was accomplished the procedure functioned well.

V. FURTHER RESEARCH

The results of this research demonstrate the applicability of interactive heuristics for transportation design. However, much additional research must be conducted before definitive conclusions can be drawn of which models should be employed and how great a role the human should play in the process.

APPENDIX A

TECHNICAL REPORTS

APPENDIX A-1

TECHNICAL REPORT:

SET PARTITIONING BASED HEURISTICS
FOR INTERACTIVE ROUTING

SET PARTITIONING BASED HEURISTICS FOR INTERACTIVE ROUTING

by

Frank H. Cullen
John J. Jarvis
H. Donald Ratliff
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332

August, 1980

This research is supported in part under Transportation Systems Center contract #DOT-TSC-1618 and Office of Naval Research Contract #N00014-79-C-0035

ABSTRACT

The set partitioning model is used as the basis for an interactive approach for solving a broad class of routing problems. A pricing mechanism is developed which can be used with a variety of methods in generating improving solutions. A version of the approach for delivery problems has been implemented via a colorgraphics display. The Human Aided Optimization procedure was tested on the standard 50-point, 75-point and 100-point test problems of Eilon, Watson-Gandy and Christofides [6]. In the case of the first two test problems, the procedure was able to generate the best known solutions. In the 100-point problem, a better solution was generated than the current best known solution.

1. INTRODUCTION

We will consider here a set partitioning based approach for solving a broad class of routing problems. The approach is designed to take advantage of a high level of human interaction; the current implementation is interactive via a colorgraphics display. However, many of the concepts discussed here could be easily implemented in an automatic system.

The routing problem which motivated much of this work is what is called the static or subscriber dial-a-ride problem. This problem will be discussed in detail in later sections. It is one of the more complex members of the class of routing problems which are amenable to the approach presented here. This class also includes many practical delivery problems.

In order to introduce the underlying methodology which provides the basis for the approach, consider a very simple delivery example. Assume that a depot is located at the square box labeled D in Figure 1. From this depot a single delivery is to be made to each of the points represented by numbered circles. The numbers on arcs connecting the circles represents the travel distance between delivery points. Assume also that each vehicle (e.g. truck) can deliver to a maximum of two points on a single trip. The objective is to determine which vehicle should deliver to each point and the routing for the vehicles which minimizes the total distance travelled.

Each column in the matrix of Table 1 represents one possible vehicle route. For example, column one represents a vehicle travelling from the depot to delivery point (1) and returning. The c_j row indicates the

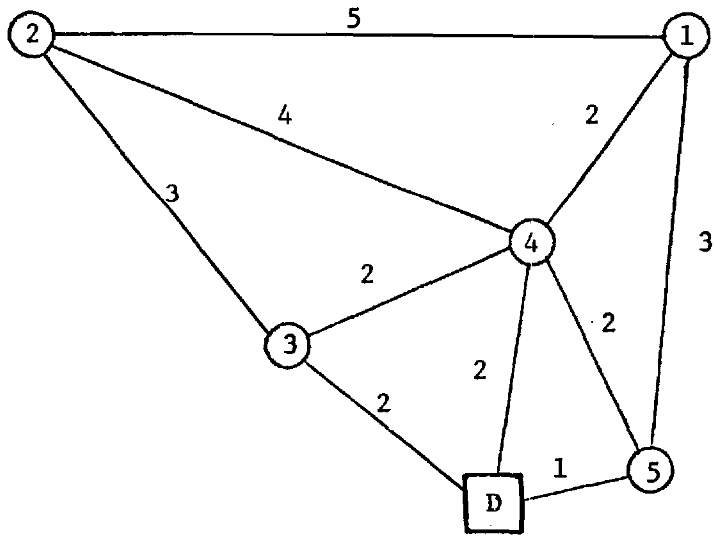


Figure 1. Delivery Example Network

route	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
c_j	8	10	4	4	2	14	10	8	8	10	11	12	6	6	5
	1	0	0	0	0	1	1	1	1	0	0	0	0	0	0
	0	1	0	0	0	1	0	0	0	1	1	1	0	0	0
	0	0	1	0	0	0	1	0	0	1	0	0	1	1	0
	0	0	0	1	0	0	0	1	0	0	1	0	1	0	1
	0	0	0	0	1	0	0	0	1	0	0	1	0	1	1

Table 1. Matrix Corresponding to Routes in the Delivery Example of Figure 1.

distance travelled for each trip. For example, column six represents a vehicle proceeding from the depot to delivery point (1), on to delivery point (2) and from there, back to the depot. The value of c_6 is 14, the total distance travelled for this trip. By enumerating each of the possibilities, as has been done for this matrix, the problem becomes one of selecting a set of columns such that every row is represented in exactly one column and the sum of the costs of the columns selected is the smallest possible. This integer program is called a "set partitioning model".

The set partitioning model was originally proposed for a similiar class of routing problems by Charnes and Miller [3] and for this specific problem by Balinski and Quandt [2]. The model is very powerful in the sense that many realistic route constraints and cost functions can be handled easily in the column enumeration process. The obvious shortcomming of the model is that there are typically a very large number of columns to be enumerated and the resulting integer program is very large. The approach presented here is heuristic in the sense that we generate only a subset of the possible columns or routes and in general we do not solve the set partitioning model to optimality.

The set partitioning model has two very desirable features for interactive optimization. The first is that any route generated can be included as a column in the model. This allows the human interactor to utilize his/her intuition and spatial perception as well as a wide spectrum of mathematical techniques to generate new routes. The second feature is that, unlike more general integer programs, a feasible solution to the set partitioning model provides the basis for pricing information which can be used to generate new candidate columns.

We will restrict the class of routing problems considered here to be those for which any subroute of a feasible route is also a feasible route with cost less than or equal to the cost of the original route. By imposing this restriction, as long as there is at least one 1 in every row of the partial set partitioning model that we have enumerated, we can easily generate feasible partitions. The only other restriction that we put on the class of routing problems is that we be able to pose them in a natural way as set partitioning problems. However, it should be noted that if there is not a nice spatial representation of the routing problem, the human interactor is much more restricted in his/her contribution.

2. SET PARTITIONING AND ROW PRICES

The set partitioning problem can be stated as

$$\text{minimize} \qquad Z = \sum_{j=1}^{n} c_j x_j \tag{1}$$

$$\text{subject to} \qquad \sum_{j=1}^{n} a_{ij} x_j = 1 \quad \text{for } i = 1,2,\ldots,m \tag{2}$$

$$x_j = 0 \text{ or } 1 \quad \text{for } j = 1,2,\ldots,n. \tag{3}$$

For the delivery example in Table 1, the set partitioning model has the second row as the value of the c_j and rows three through seven as the values of the a_{ij}. The variables which are set to one in a solution to the set partitioning problem will be called a "partition." We will denote a partition as $J^k = \{j \mid x_j^k = 1\}$. Balas and Padberg [1] provide a recent survey of results related to set partitioning problems.

A fundamental idea underlying much of the work presented here is the concept of "row prices."

<u>Definition</u>: $P^1 = (p_1^1, p_2^1, \ldots, p_m^1)$ is a set of feasible row prices corresponding to the partition J^1 if

$$\sum_{i=1}^{m} p_i^1 a_{ij} = c_j \quad \text{for } j \varepsilon J^1. \tag{4}$$

It will be useful to interpret the price p_i^1 as an estimate of the cost to satisfy constraint i using solution X^1. For the delivery problem,

p_i^1 is then an estimate of the cost of satisfying the requirement of delivery point i using the route corresponding to partition J^1.

Theorem 1: Given a set of feasible row prices $(p_1^1, p_2^1, \ldots, p_m^1)$ corresponding to partition J^1 with value Z^1, any other partition J^2 has value.

$$Z^2 = Z^1 - \sum_{j \varepsilon J2} \sum_{i=1}^{m} (p_i^1 a_{ij} - c_j) \tag{5}$$

Proof:

$$\sum_{j \varepsilon J^2} \left(\sum_{i=1}^{m} p_i^1 a_{ij} - c_j \right) = \sum_{i=1}^{m} p_i^1 \sum_{j \varepsilon J^2} a_{ij} - Z^2$$

Since J^2 is a partition, $\sum_{j \varepsilon J^2} a_{ij} = 1$ for each $i = 1,2,\ldots,m$. Also, since $(p_1^1, p_2^1, \ldots, p_m^1)$ are feasible row prices corresonding to X^1 we have $\sum_{i=1}^{m} p_i^1 = Z^1$. Hence the result follows.

Corollary 1: For any set of feasible row prices P^1 corresponding to a partition J^1 if

$$\sum_{i=1}^{m} (p_i^1 a_{ij} - c_j) \leq 0 \tag{6}$$

for $j = 1,2,\ldots,n$ then X^1 is optimum.

It can also be shown, using linear programming duality, that a set of feasible row prices P^1 satisfying Corollary 1 exists if and only if X^1 is an optimum solution with the constraints $x_j = 0$ or 1 replaced by $x_j \geq 0$ for $j = 1,2,\ldots,n$.

The quantity $\sum_{i=1}^{m} p_i^1 a_{ij} - c_j$ will be interpreted as the "potential" savings over the value of Z^1 which can result from constructing a partition that includes column j. Note from Theorem 1 that the potential savings can actually be achieved only if a partition can be constructed from columns with nonnegative potential savings.

3. POTENTIAL SAVINGS HEURISTIC

Given a partition J^1 and a corresponding set of feasible row prices P^1, an attractive heuristic for attempting to generate a better partition is the following:

Step (0): Let $J^2 = \emptyset$ (J^2 will be the indices of columns in the new partition) and $N = \{1,2,\ldots,n\}$, (N will be the indices of columns which are candidates for inclusion in J^2)

Step (1): Calculate the potential savings $\sum_{i=1}^{m} p_i^1 a_{ij} - c_j$ for $j = 1,2,\ldots,n$.

Step (2): Pick the column k in N with the largest potential savings, $\sum_{i=1}^{m} p_i^1 a_{ik} - c_k$

Step (3): For $i = 1,2,\ldots,n$, if $a_{ik} = 1$ set $a_{ij} = 0$ for all $j \neq k$. (Note from the assumption of section 1 that any subroute of a feasible route is also a feasible route, the new columns are legitimate)

Step (4): Let $J^2 = J^2 \cup \{k\}$ (i.e., put column k in the new partition) and $N = N - \{k\}$

Step (5): Delete from N all j for which $a_{ij} = 0$ for all $i = 1,2,\ldots,m$.

Step (6): If $N = \emptyset$ stop. Otherwise go to step (2).

Note that under the assumption that any subset of a route is also a feasible route (discussed in section 1) this procedure will always terminated with J^2 as a partition although not necessarily a better partition than J^1. If convenient, c_j should be recomputed whenever step (3) affects a change in column j, as the actual route may change. Computation to date indicates that an optimum or near optimum solution to the set partitioning problem is determined very quickly by repeated application of the potential savings heuristic (note that the a_{ij} should be reset to their original values after each repetition). The heuristic is repeated until either optimality is proven (i.e., for some X^k all $\sum_{i=1}^{m} p_i^k a_{ij} - c_j \leq 0$) or until some specified number of partitions has been generated.

To illustrate the potential savings hueristic, consider again the delivery example depicted in Figure 1 and Table 1. Suppose that we select $J^1 = \{1,2,\ldots,5\}$ as an initial partition. A set of feasible row prices P^1 is given in Table 2. (The question of how to generate "good" feasible row prices will be addressed in the next section.) The corresponding potential savings $\sum_{i=1}^{m} p_i^1 a_{ij} - c_j$ are also given in Table 2. Applying the potential savings heuristic and breaking ties by selecting the column with the lowest index yields the new partition $J^2 = \{6,13,5\}$. The new partition has a cost of $Z^2 = 22$ as compared to a cost $Z^1 = 28$ for the initial solution.

Using the row prices P^2 and potential savings $\sum_{i=1}^{m} p_i^2 a_{ij} - c_j$ of Table 2 and reapplying the potential savings heuristic yields the partition $J^3 = \{8,10,5\}$ which has a cost of $Z^3 = 20$. Again, from Table 2 we find that using the row prices P^3 gives potential savings $\sum_{i=1}^{m} p_i^3 a_{ij} - c_j \leq 0$ for $j = 1,2,\ldots,n$. Hence, from Corollary 1 the partition J^3 is optimum.

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
c_j	8	10	4	4	2	14	10	8	8	10	11	12	6	6	5	P^1	P^2	P^3
	1					1	1	1	1							8.0	6.2	5.3
		1				1				1	1	1				10.0	7.8	7.1
			1				1			1			1	1		4.0	3.0	2.9
				1				1			1		1		1	4.0	3.0	2.7
					1				1			1		1	1	2.0	2.0	2.0
$\sum_{i=1}^{n} p_i^1 a_{ij} - c_j$	0	0	0	0	0	4	2	4	2	4	3	0	2	0	1	$x_6 = 1$		
	–	–	0	0	0	–	-6	-4	-6	-6	-7	-10	2	0	1	$x_{13} = 1$		
	–	–	–	–	0	–	–	–	-6	–	–	-10	–	-4	-3	$x_5 = 1$		
$\sum_{i=1}^{n} p_i^2 a_{ij} - c_j$	-1.8	-2.2	-1	-1	0	0	-.8	1.2	.2	.8	-.2	-2.2	0	-1	0	$x_8 = 1$		
	–	-2.2	-1	–	0	-6.2	-7	–	-6	.8	-3.2	-2.2	-3	-1	-3	$x_{10} = 1$		
	–	–	–	–	0	–	–	–	-6	–	–	-10	–	-4	-3	$x_5 = 1$		
$\sum_{i=1}^{n} p_i^3 a_{ij} - c_j$	-2.7	-2.9	-1.1	-1.3	0	-1.6	-1.8	0	-.7	0	-1.2	-2.9	-.4	-1.1	-.3			

Table 2. An Example Illustrating the Potential Saving Heuristic with $J^1 = \{1,2,3,4,5,\}$, $J^2 = \{6,13,5\}$, and $J^3 = \{8,10,5\}$

4. ROW PRICING

For a given partition X^k a set of feasible row prices is obtained by allocating the column cost c_j for each $j \varepsilon J^k$ among the rows having $a_{ij} = 1$. For the delivery example, this corresponds to allocating the trip cost among the delivery points of the trip. When a column $j \varepsilon J^k$ contains only one $a_{ij} = 1$, the row price is $p_i^k = c_j$. However, when a column j contains more than one $a_{ij} = 1$, there are an infinite number of possible sets of prices. As an example consider the partition $J^3 = \{8,10,5\}$ for the problem in Table 2. Since column 5 has only $a_{5,5} = 1$ the value $p_5^3 = 2$ is unique. Column 8 has both $a_{1,8} = 1$ and $a_{4,8} = 1$, hence the cost $c_8 = 8$ could be allocated between rows 1 and 4 in an infinite number of ways. Similiarly, column 10 has both $a_{2,10} = 1$ and $a_{3,10} = 1$, hence $c_{10} = 10$ could be allocated between rows 2 and 3 in an infinite number of ways. If we allocate c_8 as $p_1^3 = 4$ and $p_4^3 = 4$ and allocate c_{10} as $p_2^3 = 5$ and $p_3^3 = 5$, the resulting $\sum_{i=1}^{m} p_i^3 a_{ij} - c_j$ do not indicate that J^3 is an optimum partition. Hence the set of prices P^3 given in Table 2 are clearly better since they do indicate that J^3 is on optimum partition.

Ideally, we would like a set of prices which would drive the potential savings heuristic toward an improving solution and would indicate optimality when no improving solution is possible (i.e., We would like the prices to be analogous to dual variables in linear programming). Unfortunately, it is easy to construct cases for which no such prices exist (i.e., any problem for which the integer and continuous solution differ). For the delivery problem and the more complex dial-a-ride problem (to be discussed later), allocating column cost in proportion to the cost of serving the delivery points one-at-a-time is intuitively appealing and seems to work very well. As an illustration consider again the partition J^3 in Table 2. Column 8 has $a_{1,8} = 1$ and $a_{4,8} = 1$. The cost of serving delivery point 1 if it is the only point in a trip is $c_1 = 1$. The cost of serving delivery point 4 if it is the only point in a trip is $c_4 = 4$. The prices for rows 1 and 4 were determined as $p_1^3 = \frac{c_1 c_8}{c_1 + c_4} = 5.3$ and $p_4^3 = \frac{c_4 c_8}{c_1 + c_4} = 2.7$. The other prices in Table 2 were determined similiarly.

When there are substantial differences in the amounts of vehicle capacity required by the delivery points, it seems reasonable to allocate column cost in proportion to the "weighted" cost of serving points one-at-a-time. Here the weights are the delivery sizes. For example if w_i is the vehicle capacity required by delivery point i, we specify the price of row 1 with respect to J^3 above as $P_1 = \frac{c_1 w_1 c_8}{c_1 w_1 + c_4 w_4}$. These and other pricing alternatives are currently being tested for delivery problems under various types of constraints.

There are a variety of other mechanisms for allocating the prices to the demand points. Some allocation procedures may work well in some situations while others may be suited to other problems. The human (or automatic algorithm) may wish to employ different pricing procedures as the algorithm progresses.

5. COLUMN GENERATION

For large scheduling and routing problems it is generally not practical to generate all columns of the corresponding set partitioning model. The remainder of this paper will be concerned with using information gleaned from one solution via Theorem 1 to generate a new and hopefully better solution. This is accomplished by either generating new columns, adding them to the current set partitioning model, and then resolving the model or by using the information from Theorem 1 directly to generate a new solution. In the latter case it is not necessary to retain the columns of the set partitioning model. However, if the columns are retained, it is possible to further improve the solution by periodically solving the set partitioning model.

We should note that for the class of scheduling and routing problems being considered here, it is very easy to generate an initial solution. For our examples we use the identity solution (e.g., in the delivery problem this is the solution which has each vehicle making a single delivery) as our initial solution. However, any feasible solution could be used as the initial solution.

Clearly there is a broad spectrum of possible approaches that one might use to generate new columns and/or new solutions to the set partitioning model. In fact, variations of many of the heuristics which have been applied to delivery problems can be used very effectively in conjunction with Theorem 1. In the next section we will discuss the use of the Clarke and Wright [3] savings procedure in conjunction with the delivery problem. In later sections we will discuss more complex clustering and chaining heuristics as we have applied them to the dial-a-ride problem.

6. CLARKE AND WRIGHT PROCEDURE WITH PRICING

The "savings" heuristic of Clarke and Wright [3] is the most widely known of the heuristics developed to date for delivery problems. The algorithm proceeds by calculating a savings for each pair of delivery points i and j defined as

$$s_0(i,j) = 2d_{0i} + 2d_{j0} - (d_{0i}+d_{ij}+d_{j0}) = d_{0i} + d_{j0} - d_{ij}$$

which is the savings in mileage of supplying delivery points i and j on the same route as opposed to supplying them individually directly from the depot (d_{oi} is the distance from the depot to delivery point i). Routes are then constructed either one-at-a-time or in parallel by considering pairs of points in order of decreasing savings and including them in the same route if such a route is feasible.

Suppose that we consider once more the delivery example in Figure 1 and the covering solution information in Table 2. Note that the Clarke and Wright savings values are exactly the values of $\sum_{i=1}^{m} p_i^1 a_{ij} - c_j$ in Table 2 since $d_{0i} = p_i^1/2$ for $i = 1,2,\ldots,m$. Applying the C-W savings algorithm yields the same routing configuration as $J^2 = \{6,13,5\}$. At this point, the C-W algorithm would terminate. However, suppose that we set $d_{0i} = p_i^2/2$ for $i = 1,2,\ldots,m$. The new savings are then exactly the $\sum_{i=1} p_i^2 a_{ij} - c_j$ in Table 2. Applying C-W algorithm yields the same routing configuration as $J^3 = \{8,10,5\}$ which as noted previously is the optimum solution to this delivery example. Hence, for this example at least the C-W algorithm without pricing did not yield an optimum solution while the same algorithm when combined with pricing did yield the optimum solution.

Note that when routes are allowed to contain more than two trips, only a small subset of the set partitioning columns would be generated using this procedure. Also, note that the prices and savings can be calculated without ever generating the set partitioning matrix.

When the number of points allowed in a route exceeds two, some interesting questions arise as to exactly how the algorithm should be implemented. As an illustration, suppose that in the example we allow a vehicle to deliver to at most three points rather than two. Now suppose that we start with the solution J^3 in Table 2. We see that the two-at-a-time potential savings are all non-positive. However, if we ignore this fact and proceed with the algorithm, we would put points 1 and 4 in the same route since their potential savings of zero is maximum. If we then consider adding a third point to this route, we find that adding point 5 has a potential savings of 2 which is maximum. Finally, we combine points 2 and 3 into a single route since this has a potential savings of zero. The resulting routes (4,1,5) and (2,3) has a length of 10. Therefore, in this case at least we can get better information by recalculating the potential savings after each augmentation of a route. This recalculation is not done in most implementations of the C-W algorithm.

Clearly, when constructing a route containing more than two points one must decide where in the route to put each additional point. The potential savings can be determined exactly only by solving a travelling salesmen problem over each new point which is a candidate to be added to the route. This is computationally expensive if a route can contain a large number of points. In most implementations of the C-W algorithm, new points are simply added on to the end of the route being constructed. When the algorithm is implemented interactively using computer graphics, it appears that the human can perform an important role both in selecting candidate points and in inserting them logically into routes.

7. LOCATION - ALLOCATION WITH PRICING

Another procedure which Krolak and Nelson [5] have found effective in approaching the delivery problem utilizes the location - allocation model of Cooper [4]. This basic concept can also be used in conjunction with Theorem 1 to generate intuitively appealing columns to add to the set partitioning model.

To illustrate the procedure consider the delivery example illustrated in Figure 2. Again the circles represent delivery points and the square represents the depot. The basic idea is to use a surrogate distance rather than the actual distance in determining the points which are assigned to each vehicle. The surrogate distance is obtained by assuming that the vehicle travels from the depot to a specified cluster point, represented in Figure 2 by the dashed circles. It then makes the deliveries, returning after each delivery to the cluster point. After all deliveries have been made, the vehicle returns to the depot. Under this surrogate distance, the problem becomes one of locating the cluster points, one for each vehicle, and then assigning the delivery points to each vehicle.

If (a_0,b_0) represents the coordinates of the depot, (a_i,b_i) represents the coordinates of delivery point i, (x_j,y_j) represents the coordinates of cluster point j, and assuming Euclidean distance, the problem can be modeled as follows:

$$\min_{x,y,z} \sum_{i=0}^{m} \sum_{j=1}^{n} 2[(x_j-a_i)^2 + (y_j-b_i)^2]^{\frac{1}{2}} z_{ij}$$

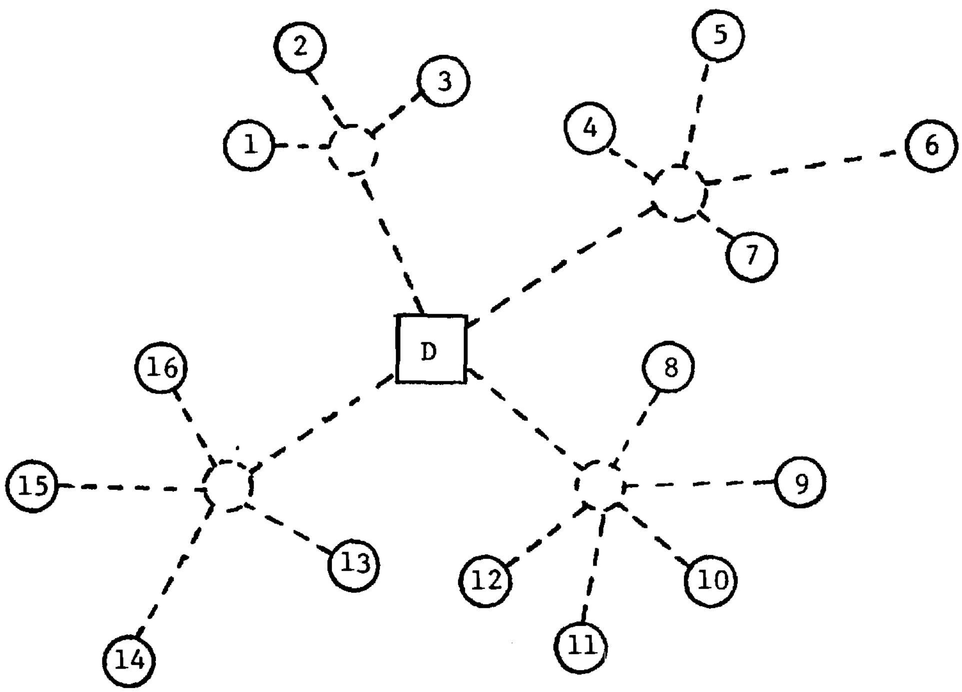


Figure 2. Delivery Example to Illustrate the Location - Allocation Model for Clustering.

$$\text{s.t.} \quad \sum_{i=1}^{m} z_{ij} \le K \qquad j = 1,2,\ldots,n$$

$$\sum_{j=1}^{n} z_{ij} = 1 \qquad i = 1,2,\ldots,m$$

$$z_{ij} = 0 \text{ or } 1 \quad \text{all } i,j$$

$$z_{0j} = 1 \qquad j = 1,2,\ldots,n$$

where n is the number of vehicles and K is the vehicle capacity. When $z_{ij} = 1$, delivery point i is assigned to vehicle j and when $z_{ij} = 0$, delivery point i is not assigned to vehicle j.

While there is no method for efficiently solving this problem optimally, the following is an attractive heuristic. Pick a set of locations for the cluster points. With these coordinates fixed, solve the resulting assignment problem. With these values of z_{ij} fixed, solve the resulting location problem. Continue alternating between the assignment and location problems for some specified number of iterations or until there is no further improvement in the objective. Once a cluster has been determined, the vehicle is then routed among the points of the cluster.

A slight modification of this model, together with Theorem 1, allows us to generate attractive new columns for the set partitioning problem. Suppose that we have a solution to the set partitioning problem and a set of row prices $p_1,p_2,\ldots,p_m$. Now consider the model

$$\min_{x,y,z} \sum_{i=0}^{m} \sum_{j=1}^{n} 2[(x_j-a_i)^2 + (y_j-b_i)^2]^{\frac{1}{2}} z_{ij} - \sum_{i=1}^{m} \sum_{j=1}^{n} p_i z_{ij}$$

$$\text{s.t.} \sum_{i=1}^{m} z_{ij} \leq K \quad j = 1,2,\ldots,n$$

$$\sum_{j=1}^{n} z_{ij} \leq 1 \quad i = 1,2,\ldots,m$$

$$z_{ij} = 0 \text{ or } 1 \quad \text{all } i,j$$

$$z_{0j} = 1 \quad j = 1,2,\ldots,n$$

Any cluster generated by this model will have a positive potential savings, with respect to the surrogate distance. Hence, it corresponds to an attractive column to add to the set partitioning problem (considering surrogate distances).

Since the second constraint has been changed to an inequality, not all delivery points will be assigned to cluster points. This simply means that the current row price for the delivery point is more attractive than the cost of serving the delivery point in alternative clusters considered by the model. The model can be solved using the same heuristic discussed for the earlier location - allocation model.

8. DIAL-A-RIDE PROBLEM

The dial-a-ride problem is a much more complex routing problem than the delivery problem. In the dial-a-ride problem we are given an origin-destination trip matrix and an underlying network on which the trips are to be made. There is a single item (people, goods, etc.) (demand for service) at each origin that needs to be transported to its specified destination. The items are transported from origins to destinations on vehicles each having capacity K. We wish to satisfy the trip requirements while travelling the minimum distance.

This is a "static" version of the dial-a-ride problem since time is not considered. There are a number of more complex versions of this problem, but this version is sufficient to demonstrate the basic ideas of our approach.

The set partitioning model for the dial-a-ride problem is analogus to that of the delivery problem, but here rows represent trips rather than simple delivery points. The vehicle capacity constraints are handled by generating only routes which satisfy them.

9. DECOMPOSITION

In delivery problems, representative of the "one-ended" class of routing and scheduling problems, we need only be concerned with a single stop for a vehicle to satisfy a particular demand for service. In contrast, the "two-ended" class, which includes dial-a-ride problems, requires two stops in a specific order (sequence) to satisfy a specific demand for service. It is this requirement for sequencing of "pickup" and "dropoff" point pairs which adds greatly to the difficulty of handling the two-ended class of vehicle routing problems.

When one considers examples of two-ended problems, it becomes immediately apparent that the sequencing requirements greatly inhibit the complex pattern processing abilities of the human. Figure 3 illustrates an example of a 25 trip dial-a-ride problem. Rather than displaying order and structure, the problem resembles so much spaghetti. It is clear that for such problems the human interactor needs more help in generating good columns for the set partitioning model.

Unfortunately, it is also more difficult to apply straight forward methods such as the savings approach discussed earlier for the delivery example. In generating a dial-a-ride route one must be concerned with where both the origin and the destination occur in the sequence in order to calculate the potential savings. In addition, the capacity constraint may negate what otherwise appears to be good positions for the origin and destination in the sequence.

Because of this complexity, it is helpful to "decompose" the problem into two levels which we call "clustering" and "chaining". In essence, we consider a route to be made up of two components. Clusters

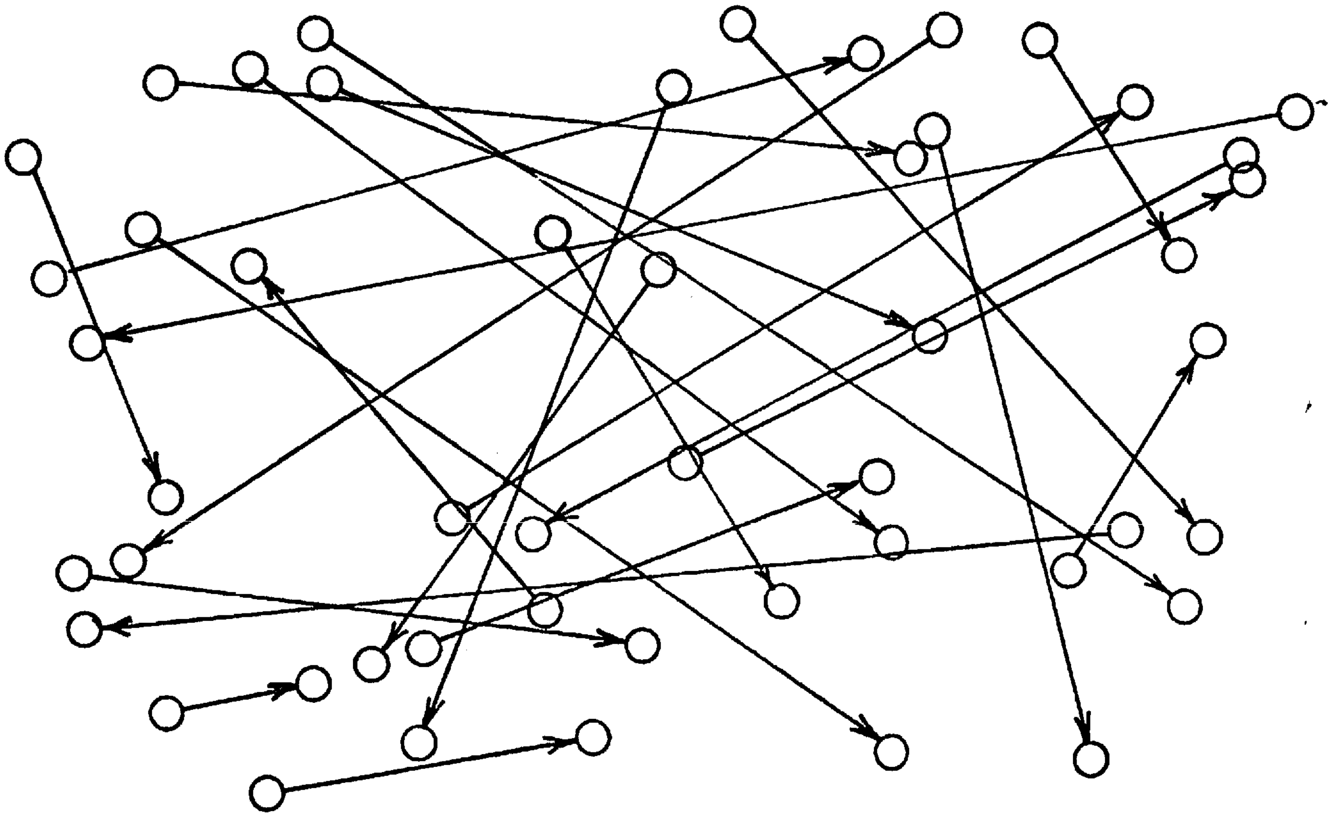


Figure 3. An Example of a 25 Trip Dial-a-Ride Problem

correspond to trips which can all be on a vehicle at one time, while chains correspond to movement from the end of one cluster to the beginning of the next. Figure 4 provides an example of five good clusters, while Figure 5 illustrates one way in which these five clusters might be linked into two chains.

The partitioning model and pricing concepts can be effectively exploited to aid in generating improving clusters and chains in a column generation approach to solving two-ended vehicle routing problems. One partitioning model can be utilized in generating improving clusters while another partitioning model helps identify better chains.

In the partitioning model for clustering, the columns represent clusters and the rows represent the individual trips (demand for service). We shall demonstrate, in the next section, just how the pricing information from the partitioning problem can be used to generate additional clusters.

In the partitioning model for chaining, the columns represent chains and the rows represent the individual trips. The function of the chaining partitioning model is to combine the clusters into good vehicle routes.

The overall solution procedure consists of linking the clustering models and chaining models together in an interactive manner. The clustering models, partitioning matrix and pricing information are used to generate good clusters. These clusters are then passed to the chaining phase where they are linked together via the chaining models, partitioning matrix and associated pricing information. After chaining, it is possible to return to the clustering phase to identify additional clusters.

The next two sections discuss the specifics of clustering and chaining

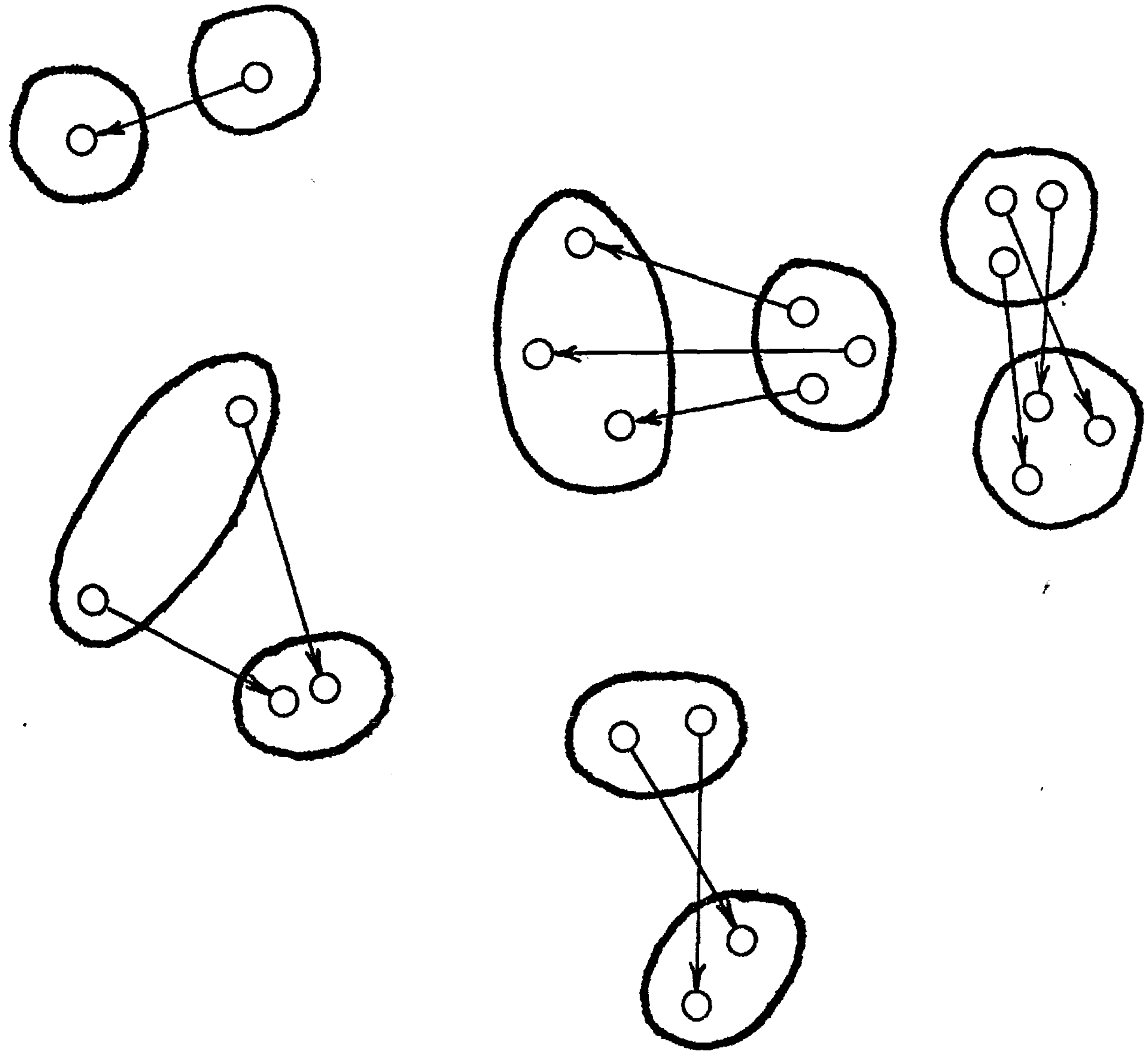


Figure 4. Example of Five Clusters

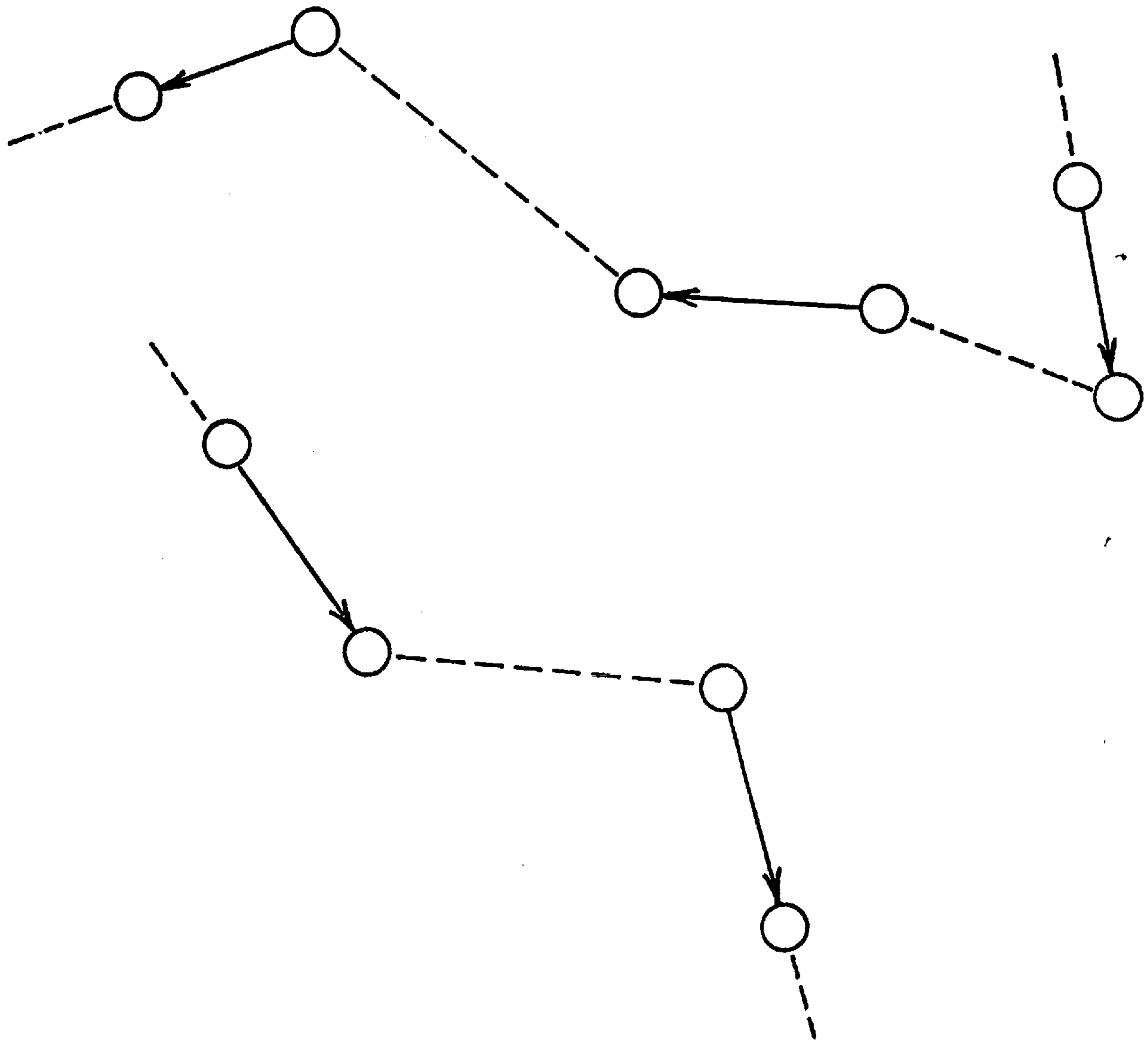


Figure 5. Example of Two Chains Linking the Five Clusters of Figure 4

10. CLUSTERING

In the previous section we introduced the clustering concept. In this section we shall provide more details of the concept as well as the structure and operation of various clustering models. Figure 6 depicts a typical cluster (in this case, three trips).

Clustering makes sense if the origins are reasonably close together and the destinations are also close together. One way to develop an evaluation of such circumstance is to locate the centroid of the origins, the centroid of the destinations and accumulate the resulting distances from the original trips. In Figure 6 we could evaluate the distances represented by (a+b+c) + (d+e+f). If this sum is small then it would make sense to cluster the trips.

By utilizing surrogate distances we lose the actual route distance evaluation; however, we gain the ability to evaluate large numbers of cluster possibilities conveniently and simultaneously. In Figure 6 we might employ Euclidean distances. In this case we could compare the sum of the row prices, $p_1 + p_2 + p_2$, generated in the covering model for clustering to the quantity 2(a+b+c) + 2(d+e+f) + g to determine whether clustering is appropriate. We can think of the latter quantity as a surrogate for the vehicle routing distance.

We can develop a straight forward extension of the Location - Allocation model discussed in Section 7 to identify good clusters for the dial-a-ride problem. In place of the quantity

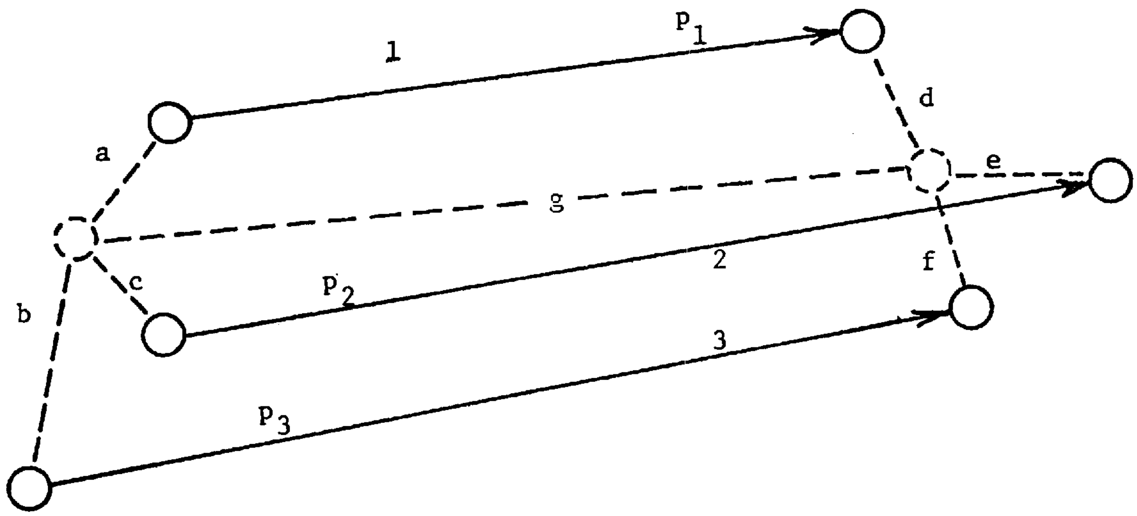


Figure 6. Surrogate Distances for a Typical Cluster

$$\sum_i \sum_j 2[(x_j - a_i)^2 + (y_j - b_i)^2]^{\frac{1}{2}} z_{ij}$$

we employ a quantity

$$\sum_i \sum_j 2[(\bar{x}_j - \bar{a}_i)^2 + (\bar{y}_j - \bar{b}_i)^2] z_{ij}$$

$$+ \sum_i \sum_j 2[(\hat{x}_j - \hat{a}_i)^2 + (\hat{y}_j - \hat{b}_i)^2]^{\frac{1}{2}} z_{ij}$$

$$+ \sum_j [(\bar{x}_j - \hat{x}_j)^2 + (\bar{y}_j - \hat{y}_j)^2]^{\frac{1}{2}}$$

where $\bar{x}_j$, $\bar{y}_j$, $\bar{a}_i$ and $\bar{b}_i$ correspond to origins and $\hat{x}_j$, $\hat{y}_j$, and $\hat{a}_i$ $\hat{b}_i$ correspond to destinations of clusters and trips.

We can also develop a savings approach to clustering in a similar manner to that described earlier for the delivery problem. As in the delivery problem, the clusters generated by the Location - Allocation models by the savings approaches or by the human interaction can be achieved in a set partitioning model. This model can then be solved to determine a "best" set of clusters.

11. CHAINING

Upon termination of the clustering process, a number of reasonably good clusters are available. This set includes not only the "best" cluster sets as selected by the covering model for clustering, but also a number of other clusters which might be nearly as good, but which were not selected in the optimal solution to the covering model. The pricing mechanism can again be utilized to select these additional "good" clusters. Figure 7 illustrates a set of clusters which might result from the clustering process. The clusters in the figure represent only the optimal solution to the clustering process. In addition, we might have other clusters available, e.g., a cluster containing only trips 1 and 2.

The clusters obtained represent good possibilities for segments (legs) of a vehicle route. The next step in the process is to link ("chain") these clusters into complete vehicles routes. Figure 8 illustrates the chaining concept. Trips 1 and 2 form one cluster, while trips 3, 4 and 5 form another cluster (see Figure 8a). In Figure 8b, trips 1 and 2 are replaced by a single pseudo (cluster) trip, as is also the case for trips 3, 4 and 5. These cluster trips are then chained together.

The interpretation of chaining is that a single vehicle will service the first set of trips (in Figure 8b these would be trips 1 and 2) and then proceed to service the next set of trips (i.e., trips 3, 4 and 5) in the chain. Figure 9 illustrates the likely vehicle route to service the two clusters.

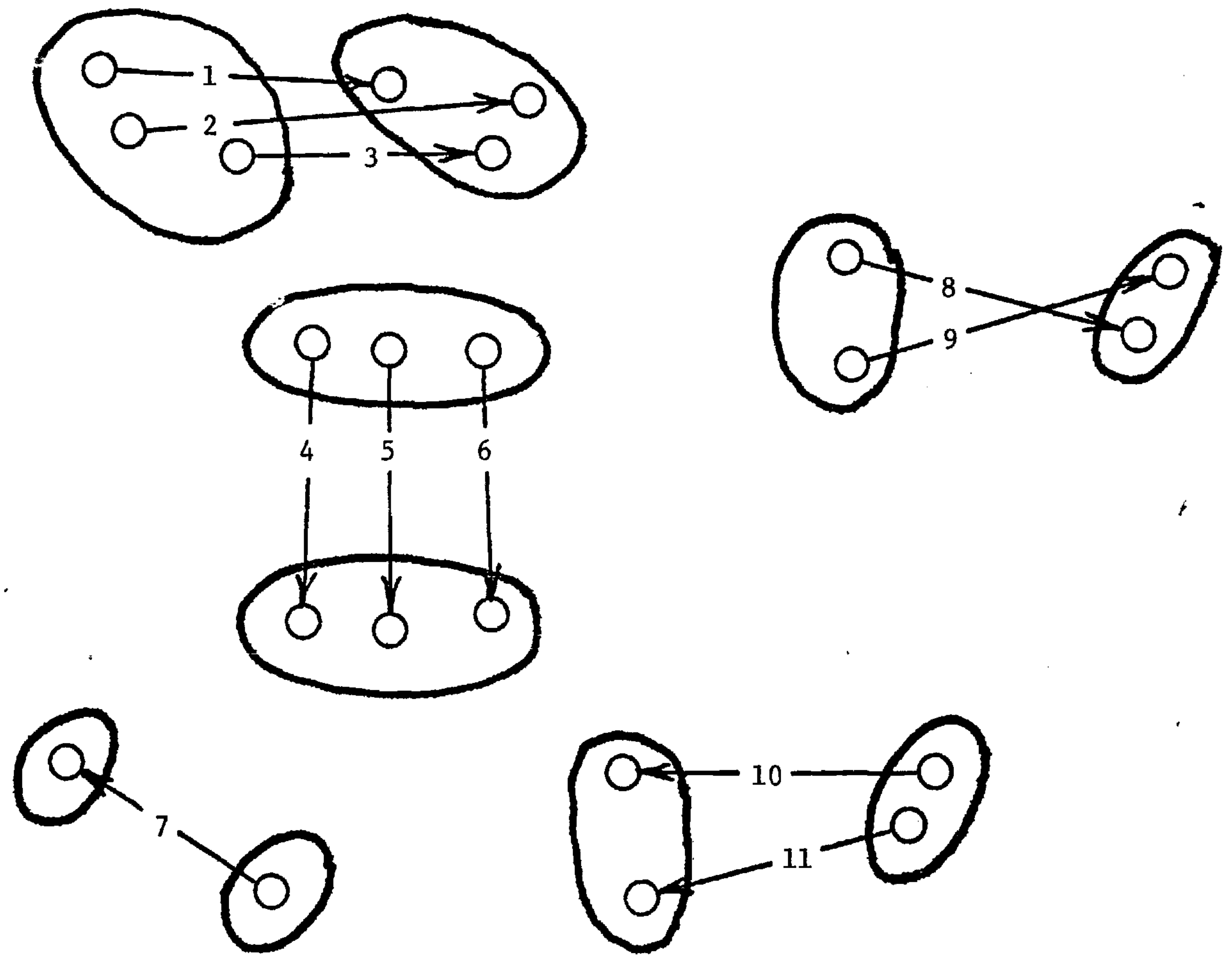
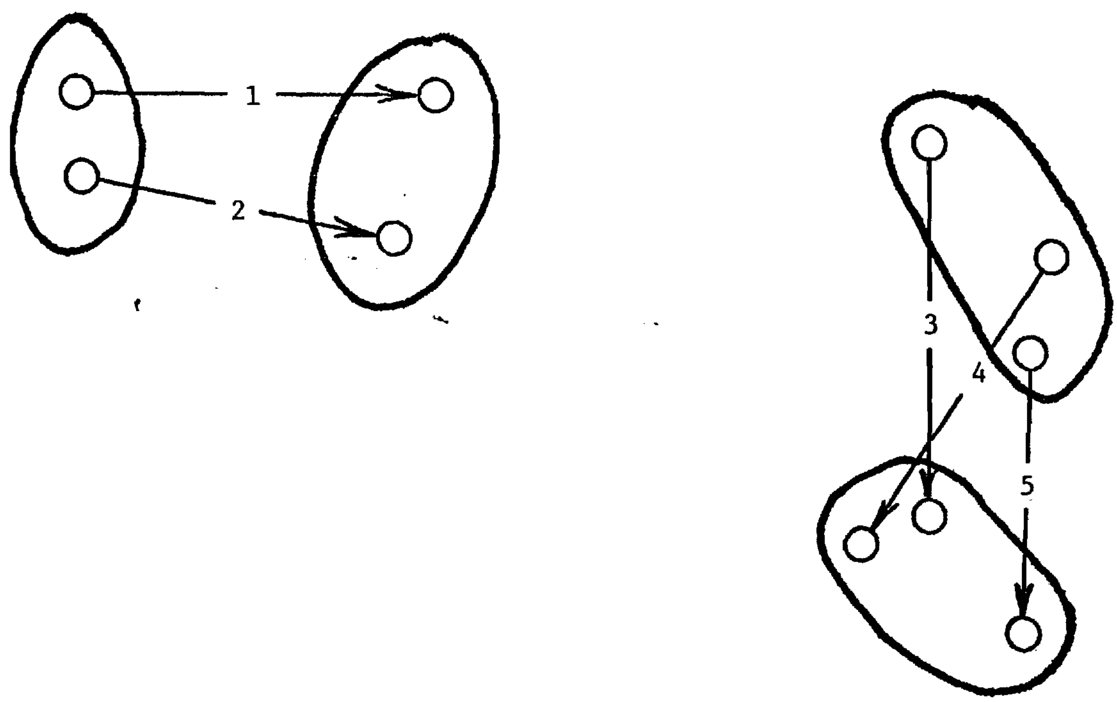
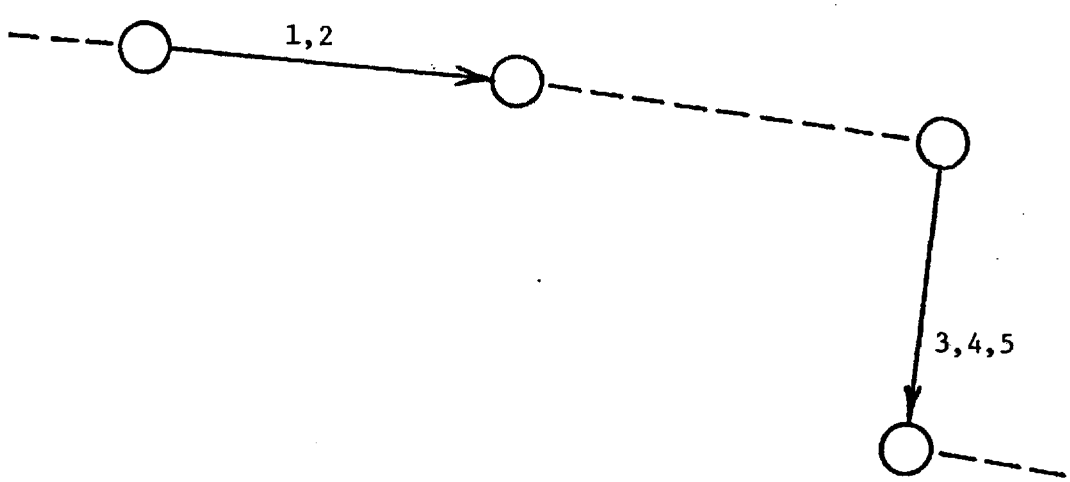


Figure 7. Example of Clusters Resulting from the Clustering Process



a. Two Typical Clusters



b. The Associated Cluster Arcs Chained Together

Figure 8. An Example of Chaining Clusters Together

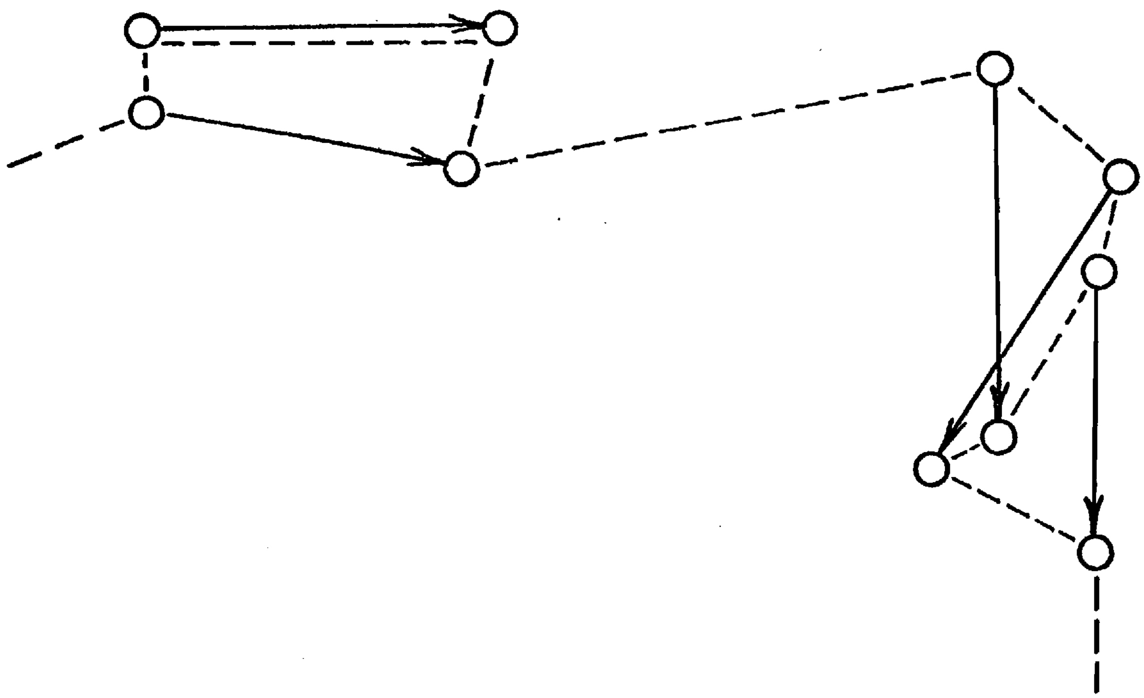


Figure 9. Probable Vehicle Route
for the Chain of Figure 8b

In the partitioning model for chaining, we associate a column of the partitioning matrix with each feasible chain (vehicle route) and a row for each trip. Column j contains 1 in row i if trip i is serviced by chain (vehicle route) j. Otherwise, it contains a 0 (zero). The zero-one variable associated with the columns provide indications of which chains were selected in the optimal partitioning solution. Table 3 illustrates a partitioning matrix for several chains in Figure 7.

The column generation process for the chaining problem is similiar to that for the delivery problem in section 5. However, there are two differences which must be addressed. The first is that in the delivery problem we were routing through "points" while in chaining, we are routing through "lines" (one for each cluster). This requires only minor changes in the savings approach; however, the Location - Allocation is no longer appropriate. The second difficulty occurs because we do not want two clusters in the same chain if they have a trip in common. This is easily avoided in the same manner as we avoid exceeding vehicle capacity in the delivery problem.

	1 clusters 1 & 2	2 clusters 1 & 4	3 clusters 2 & 5	4 clusters 1, 2 & 5	5 clusters 3, 4 & 5	6 clusters 1, 2 & 4	7 clusters 1, 2, 3 & 5
1	1	1		1		1	1
2	1	1		1		1	1
3	1	1		1		1	1
4	1		1	1		1	1
5	1		1	1		1	1
6	1		1	1		1	1
7					1		1
8		1			1	1	
9		1			1	1	
10			1	1	1		1
11			1	1	1		1

Table 3. An Example Partitioning Model
for Certain Chains in Figure 7

12. DIAL-A-RIDE SOLUTION PROCEDURE

We may put all of the foregoing ideas together into an algorithm for the dial-a-ride problem. Figure 10 provides a flowchart for the algorithm. All of the operations 1 thru 7, in Figure 10 have been previously discussed except operation 4.

The chaining procedure forms clusters into good vehicle routes. In order to increase the flexibility of the chaining procedure we must provide it with a number of clusters to work with. Thus, in addition to passing the optimal set of clusters to the chaining procedure we should also pass a number of other good clusters. For example, we might pass clusters which priced out near optimal in the partitioning problem for clustering. This is what is suggested in operation 4.

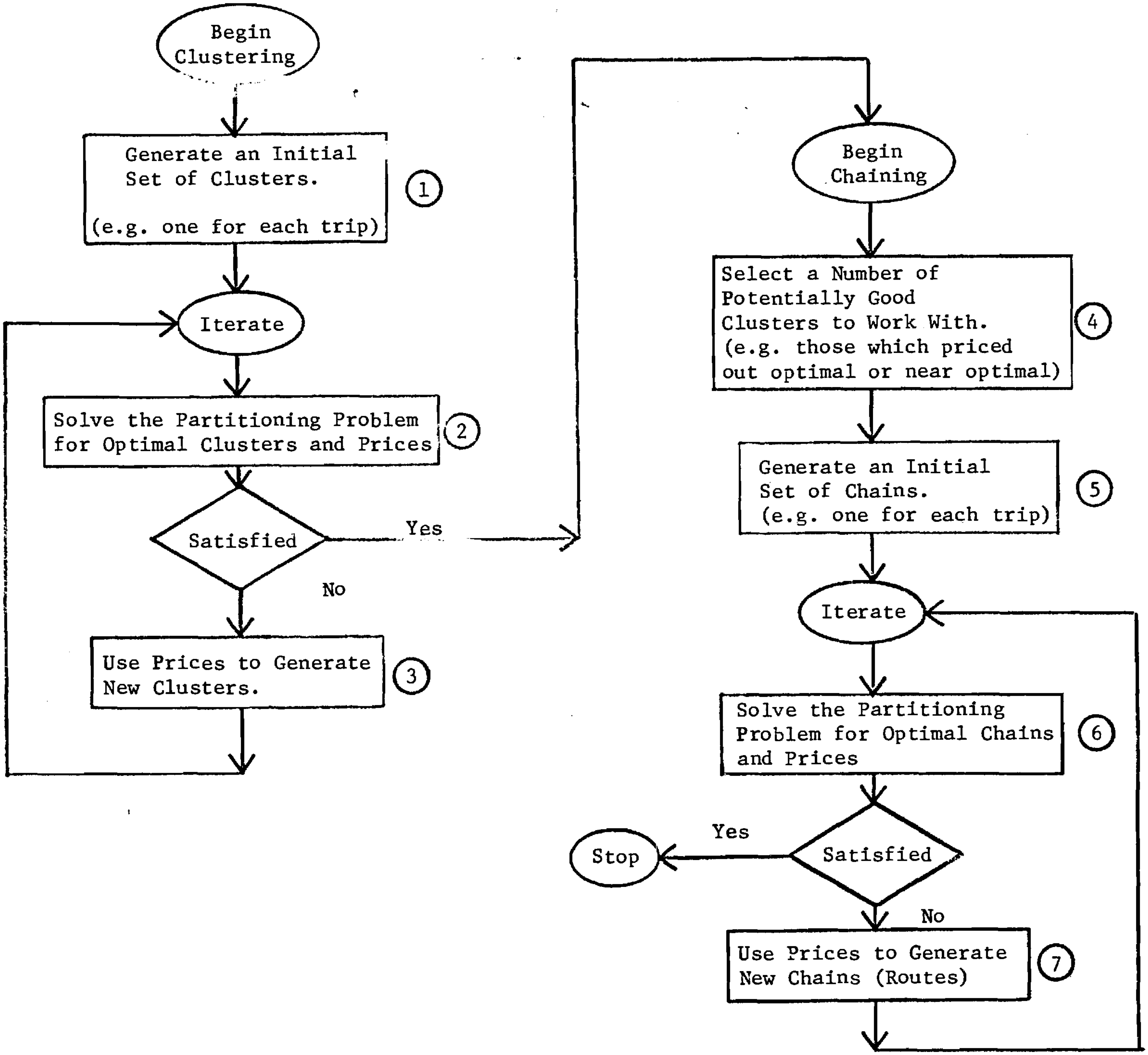


Figure 10. Flowchart of the Dial-A-Ride Solution Procedure

13. PRELIMINARY COMPUTATIONAL RESULTS

The delivery algorithm described in this paper has not yet been fully implemented. However, for evaluation purposes a version was tested which employed prices to aid the human in locating improving delivery solutions, but did not include a partitioning problem to locate a best set from the candidate routes generated. The version was tested on the 50-point, 75-point and 100-point problems of Eilon, Watson-Gandy and Christofides [6]. All of these problems (1) employ a single depot, (2) are Euclidean, and (3) have a limited vehicle capacity.

The current best known solution values for each of the three problems selected are given in Russell [8]. They are 524, 854, and 833 respectively. For each problem, one of the authors employed the interactive procedure until no improvement could be made. In every case, the best known solution value was equalled or exceeded. Table 4 presents the results of the three tests. In some cases an iteration consisted of the complete regeneration of new routes, whereas, in other cases it consisted of the exchange of demand points on a few routes.

It is very encouraging that the proposed procedure was able to generate a better solution to the 100-point problem than the current best known solution.

METHOD / PROBLEM	RUSSELL	PROPOSED	
		COST	ITERATIONS
50 POINT	524	524	5
75 POINT	854	854	4
100 POINT	833	827	11

Table 4. Results of the Proposed Method Compared With Current Best Known Solutions

14. CONCLUSIONS

An interactive delivery system and dial-a-ride system based on the concepts presented here have been implemented on a Chromatics colorgrahics terminal interfaced with a CYBER 74 mainframe computer. Preliminary tests indicate that the interactive procedure is able to compete with state-of-the-art automatic procedures for delivery problems without a great degree of human effort. Although the systems are in many ways very rudimentary, they dramatically indicate the potential for this kind of human aided optimization.

The interactive procedures should derive their greatest benefits in more complex problems containing unusual side constaints (e.g. multiple depots, time, etc.) which the human is able to perceive and control. We are in the process of making extensive modifications in the software and are testing a variety of new models and heuristics to aid in clustering and in route generation.

ACKNOWLEDGMENT

This work was partially supported by the Transportation Systems Center and the Office of Naval Research.

David J. Friedman and Robert T. Lewis contributed significantly to the implementation of the interactive dial-a-ride system. The extension of the location - allocation model to the dial-a-ride problem is due to Robert T. Lewis.

14. REFERENCES

1. Balas, E. and Padberg, M. W., "Set Partitioning: A Survey." SIAM Review, 18, (1976) 710-760.

2. Balinski, M. L. and Quandt, R. E., "On An Integer Program for a Delivery Problem," Operations Research, 12 (1964) 300-304.

3. Charnes, A. and Miller, M. H., "A Model for the Optimal Programming of Railway Freight Train Movements," Management Science, Vol. 3., No. 1, (1956) 74-92.

4. Clarke, G. and Wright, J. W., "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points," Operations Research, 12 (1964) 569-581.

5. Cooper, L., "N-Dimensional Location - Allocation Used for Cluster Analysis," Report No. COO-1493-23, Washington University, St. Louis, MO, 1969.

6. Eilon, S., Watson-Gandy, C. D. T., and Christofides, N., Distribution Management, Hafner Publishing Company, New York (1971) 200-202.

7. Krolak, P. D. and Nelson, J. H., "A Family of Truck Load Clustering (TLC) Heuristics for Solving Vehicle Scheduling Problems," Technical Report 78-2, Computer Science, Vanderbilt University, Nashville, TN, 1978.

8. Russell, R. A., "An Effective Heuristic for the M-Tour Travelling Salesman Problem with Some Side Constraints," Operations Research, 25 (1977) 517-524.

APPENDIX A-2

TECHNICAL REPORT:

CLUSTERING IN THE
DIAL-A-RIDE VEHICLE ROUTING PROBLEM

Report Series #J-80-15

CLUSTERING IN THE DIAL-A-RIDE VEHICLE ROUTING PROBLEM

BY

John J. Jarvis
Robert T. Lewis
H. Donald Ratliff

School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332

August 1980

ABSTRACT

Two models of clustering are developed for the dial-a-ride problem. End Point Clustering attempts to assign the trips to cluster points located in the two dimensional space of trips. Linear Clustering assigns the trips to lines in the trip space.

Each cluster method is fully developed methodologically and an algorithm is also presented for implementation of the techniques. Test examples as well as Rochester dial-a-ride data offer encouraging evidence of the usefullness of the cluster methods.

1. INTRODUCTION

Consider the following class of two-ended routing problems: For a given origin-destination trip matrix and an underlying network on which the trips are to be made, there are items (people, goods, etc.) at each origin that must be transported to their destination. The items are to be transported from origins to destinations on vehicles with a capacity of k in such a manner so as to minimize the total distance travelled. The particular routing problem which will be considered in this paper is the dial-a-ride problem in which a trip represents a demand for service by a subscriber of the transportation system. See Cullen, Jarvis, and Ratliff [1] for further discussion of this problem.

Because of the complex nature of this class of routing problems, it is extremely difficult to solve even moderately sized problems optimally. Instead, the approach that will be taken here is to first cluster the trips together into groups of size k or less and then to determine a minimum distance route through the origins and destinations of each cluster. Clearly any grouping of size k or less will satisfy the capacity requirements of the vehicles; however, if the clusters are chosen poorly, the resulting routes will not be of minimum distance. Furthermore, the routing problem is restricted in that the origin of each trip must be visited before the destination. Thus, the solution of a dial-a-ride problem will proceed in two steps, a clustering step and a routing step.

2. CLUSTERING

The purpose of clustering is to group together those trips which form minimum distance routes. In general, it is difficult to determine the exact contribution of each trip to the total route distance. Hence, clustering models use surrogate measures instead. These measures are composed of two parts: the distance between the origin (destination) of a trip and the cluster, and the distance along the cluster. (The details of these distances will be specified later.) The surrogate measure for each vehicle approximates the total distance that the vehicle must travel in routing through the trips assigned to it. For example, in one type of clustering the vehicle can always begin at an origin point for the cluster, go out and back to the origin of each trip, travel to a destination point for the cluster, and go out and back to the destination of each trip. Furthermore, the out-and-back distance for each trip can be weighted by a distance multiplier to improve the quality of the surrogate measure.

Two types of distance criteria can be used in clustering: Minimum Load Distance and Minimum Vehicle Distance. Load distance refers to the total of vehicle load times vehicle distance travelled whereas vehicle distance refers only to the total vehicle distance travelled. The appropriateness of either criteria depends upon the application. For example, if the routing problem concerns people, it may be more appropriate to use the load distance criteria. (Presumably people prefer to minimize their vehicle distance travelled). If the routing problem concerns objects, it may be more appropriate to use the vehicle distance criteria.

In some applications it may not be possible to "split" loads. This restriction requires that all of the load for a trip be carried on the same vehicle. In some applications it may be appropriate to split loads in terms of some unit of divisibility. For example, if people are being carried on the trips, then it does not make sense to divide into units smaller than one person.

The clustering problem to be solved in the dial-a-ride application can be formally stated as follows:

$$\text{Minimize:} \quad \sum_i \sum_j e_{ij} x_{ij} + \sum_j f_j$$

$$\text{Subject to:} \quad \sum_i x_{ij} \leq k_j \qquad \forall\ j$$

$$\sum_j x_{ij} = \ell_i \qquad \forall\ i$$

$$x_{ij} = 0, 1, \ldots, \ell_i \qquad \forall\ i, j$$

where: e_{ij} = the distance between the origin of trip i and cluster j plus the distance between the destination of trip i and cluster j.

f_j = the distance along the cluster line for vehicle j

k_j = the capacity of vehicle j

ℓ_i = the load of trip i

λ = the distance multiplier

This problem employs the minimum vehicle distance criteria with load splitting. The x_{ij} variables are in terms of units of load and determine how much of trip i's load is assigned to vehicle j. The constraint set is that of a transportation problem, and as such it can be efficiently solved.

Consider the clustering problem. So far, no mention has been made of the specific method of determining the quantities e_{ij} and f_j. Rather, e_{ij} was identified only as the distance between the origin and the destination of trip i and the cluster j, and f_j only as the distance along the cluster j. Two general methods of determining these values will be discussed.

3. ENDPOINT CLUSTERING

The first method of clustering is known as endpoint clustering and is illustrated in Figure 1. (For purposes of illustration all distances are euclidean, although it is possible to use squared euclidean and rectilinear distances as well.) In this example $e_{ij} = p_i + q_i$ or more precisely:

$$e_{ij} = ((\bar{a}_j - \bar{x}_i)^2 + (\bar{b}_j - \bar{y}_i)^2)^{\frac{1}{2}} + ((\hat{a}_j - \hat{x}_i)^2 + (\hat{b}_j - \hat{y}_i)^2)^{\frac{1}{2}}$$

Also, $f_j = r_j$ or more precisely:

$$f_j = ((\bar{a}_j - \hat{a}_j)^2 + (\bar{b}_j - \hat{b}_j)^2)^{\frac{1}{2}}$$

For this method of calculating e_{ij} and f_j, it can be shown that the objective function is not convex. Hence, it may not be possible to determine the optimal solution to this problem without resorting to some sort of enumerative procedure. Instead, a heuristic approach to solving this problem has been developed. Initial computational experience with this algorithm indicates that it converges rapidly to a locally optimal solution.

The heuristic algorithm is based on the following decomposition of the clustering problem: Suppose initial values for the cluster points $\bar{a}_j$, $\bar{b}_j$, and $\hat{a}_j$, $\hat{b}_j$ are known. (Two methods of determining these values will be discussed later.) Then it is possible to compute values for e_{ij} and f_j as shown above. Furthermore, the quantities e_{ij} and f_j become constants in the objective function and the only variables are the x_{ij}s. The problem is now a transportation problem and can be readily solved to

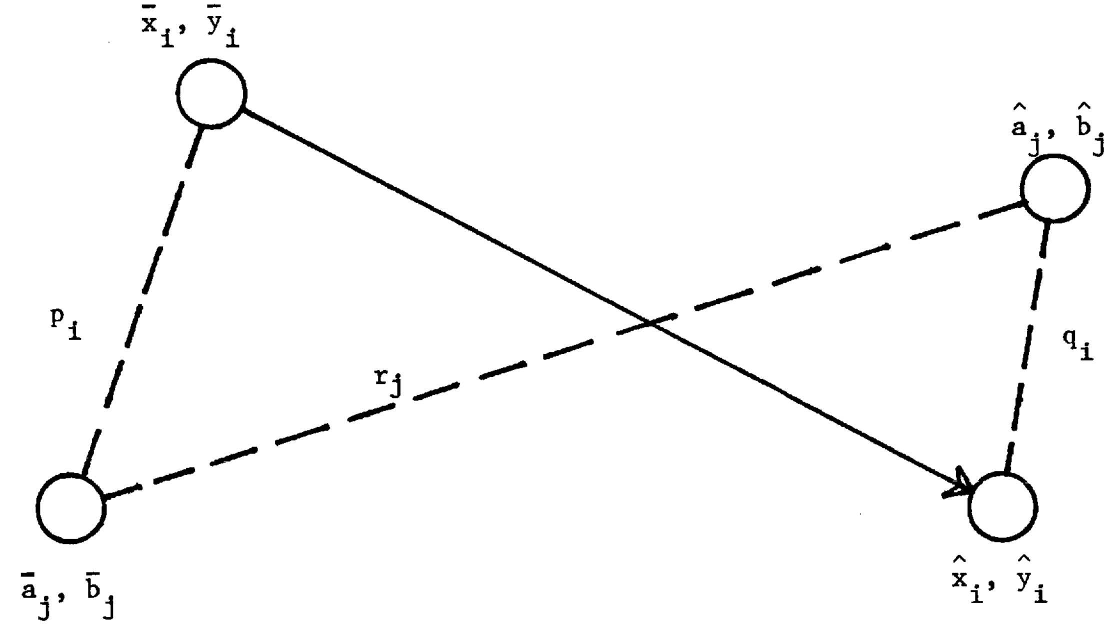


Figure 1 - Endpoint Clustering

yield locally optimal values of the x_{ij}s. With these values of the x_{ij}s as constants in the objective function, the resulting problem is now a location problem and can be readily solved using the Hyperbolic Approximation Procedure [2] to yield locally optimal values of $\bar{a}_j$, $\bar{b}_j$ and $\hat{a}_j$, $\hat{b}_j$. This decomposition is illustrated in Figure 2.

The algorithm alternates back and forth between solving a transportation problem and a location problem until no further improvement in the objective function can be obtained. Since the resulting solution is only locally optimal, the quality of the final solution is a function of the initial values of the cluster points. Two methods of determining the initial cluster points have been tested. In the first method, the origin and destination points for the clusters are set equal to the origin and destination points for selected trips. These trips are chosen so as to give a fair representation of the dispersion of trips throughout the space. In the second method, the initial cluster points are selected by an operator using a color graphics terminal on which the trips are displayed. This method attempts to exploit the ability of a human operator to recognize spatial relationships.

Location/Allocation Problem (Euclidean Distance)

$$\text{Minimize:} \quad \sum_i \sum_j [\lambda x_{ij}((\bar{a}_j - \bar{x}_i)^2 + (\bar{b}_j - \bar{y}_i)^2)^{1/2} + \lambda x_{ij}((\hat{a}_j - \hat{x}_i)^2 + (\hat{b}_j - \hat{y}_i)^2)^{1/2}] + \sum_i \sum_j x_{ij}((\bar{a}_j - \hat{a}_j)^2 + (\bar{b}_j - \hat{b}_j)^2)^{1/2}$$

$$\text{Subject to:} \quad \sum_i x_{ij} \leq k_j \qquad \forall\ j$$

$$\sum_j x_{ij} = \ell_i \qquad \forall\ i$$

$$x_{ij} = 0, 1, \ldots, \ell_i \qquad \forall\ ij$$

Transportation Problem (Euclidean Distance)

$$\text{Minimize:} \quad \sum_i \sum_j x_{ij} d_{ij}$$

Subject to: As above

$$\text{Where:} \quad d_{ij} = \lambda((\bar{a}_j - \bar{x}_i)^2 + (\bar{b}_j - \bar{y}_i)^2)^{1/2} + \lambda((\hat{a}_j - \hat{x}_i)^2 + (\hat{b}_j - \hat{y}_i)^2)^{1/2} + ((\bar{a}_j - \hat{a}_j)^2 + (\bar{b}_j - \hat{b}_j)^2)^{1/2}$$

Location Problem (Euclidean Distance)

$$\text{Minimize:} \quad \sum_i \sum_j [\lambda x_{ij}((\bar{a}_j - \bar{x}_i)^2 + (\bar{b}_j - \bar{y}_i)^2)^{1/2} + \lambda x_{ij}((\hat{a}_j - \hat{x}_i)^2 + (\hat{b}_j - \hat{y}_i)^2)^{1/2} + x_{ij}((\bar{a}_j - \hat{a}_j)^2 + (\bar{b}_j - \hat{b}_j)^2)^{1/2}]$$

Figure 2 - Decomposition Procedure for Endpoint Clustering

4. LINEAR CLUSTERING

The second method of clustering is known as linear clustering and is depicted in Figure 3. (For purposes of illustration all distances are euclidean.) The line connecting $\bar{a}_j$, $\bar{b}_j$ with $\hat{a}_j$, $\hat{b}_j$ is known as the cluster line. The point $\bar{c}_{ij}$, $\bar{d}_{ij}$ represents the point on the cluster line that is closest to the origin point for trip i. Likewise, the point $\hat{c}_{ij}$, $\hat{d}_{ij}$ represents the point on the cluster line that is closest to the destination point for trip i. (As noted in the figure $\bar{c}_{ij}$, $\bar{d}_{ij}$ can be represented as a convex combination of $\bar{a}_j$, $\bar{b}_j$ and $\hat{a}_j$, $\hat{b}_j$. Likewise, for $\hat{c}_{ij}$, $\hat{d}_{ij}$. This representation will be useful later.) In this example, $e_{ij} = p_i + q_i$ or more precisely:

$$e_{ij} = ((\bar{c}_{ij} - \bar{x}_i)^2 + (\bar{d}_{ij} - \bar{y}_i)^2)^{\frac{1}{2}} + ((\hat{c}_{ij} - \hat{x}_i)^2 + (\hat{d}_{ij} - \hat{y}_{ij})^2)^{\frac{1}{2}}$$

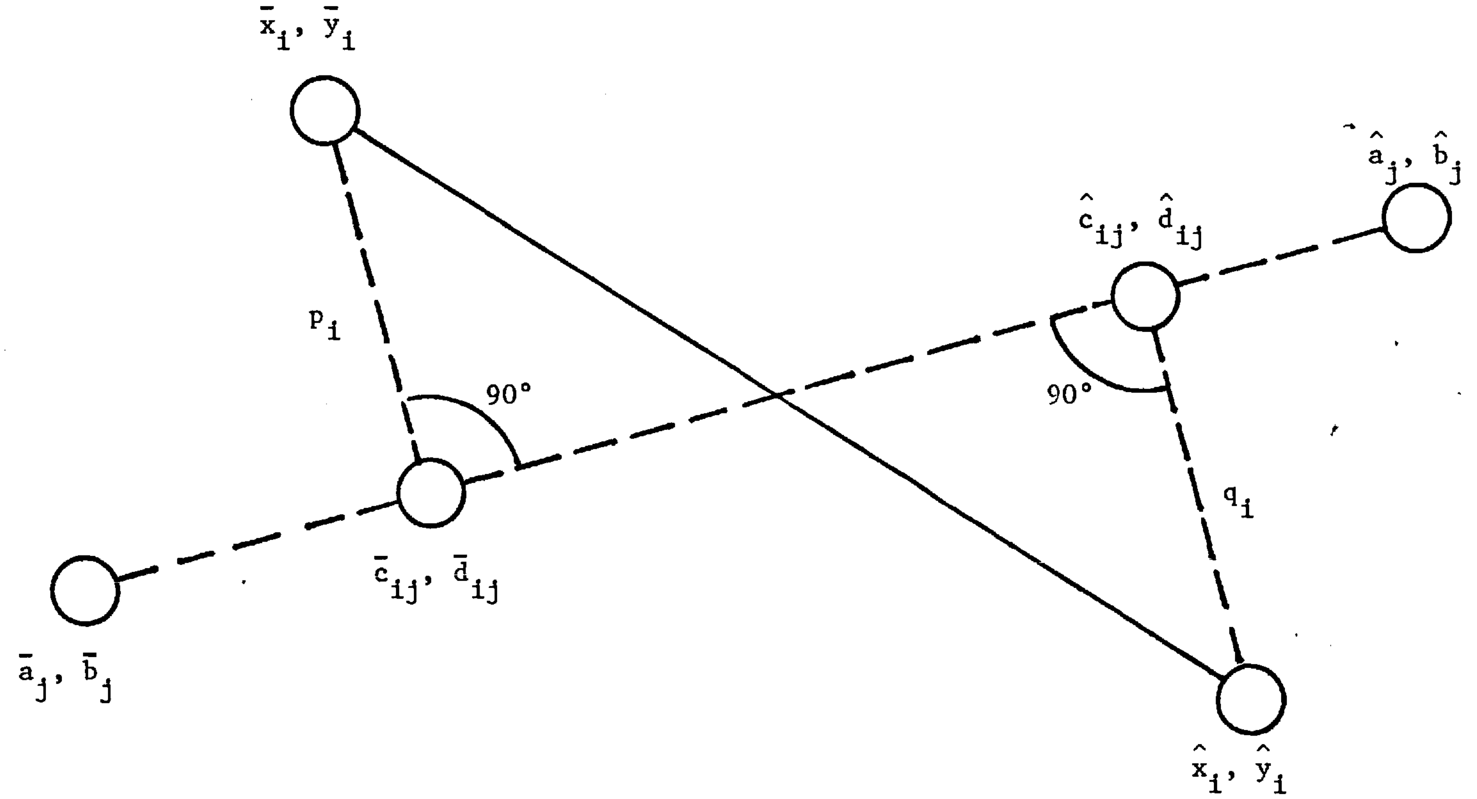
The calculation of f_j is somewhat more complicated; in terms of the notation it is as follows:

$$f_j = ((\bar{c}_{kj} - \hat{c}_{\ell j})^2 + (\bar{d}_{kj} - \hat{d}_{\ell j})^2)^{\frac{1}{2}}$$

where: $\bar{c}_{kj}$, $\bar{d}_{kj}$ is that point for which $\lambda_k = \max_i (\lambda_i)$

$c_{\ell j}$, $d_{\ell j}$ is that point for which $\pi_\ell = \max_i (\pi_i)$

Thus, f_j represents the distance between the most extreme point $\bar{c}_{ij}$, $\bar{d}_{ij}$ in the direction of the origin and the most extreme point $\hat{c}_{ij}$, $\hat{d}_{ij}$ in the direction of the destination. This is illustrated for an example involving two trips in Figure 4.



$$\bar{c}_{ij} = \lambda_i \bar{a}_j + (1 - \lambda_i)\hat{a}_j \qquad \hat{c}_{ij} = \pi_i \hat{a}_j + (1 - \pi_i)\bar{a}_j$$

$$\bar{d}_{ij} = \lambda_i \bar{b}_j + (1 - \lambda_i)\hat{b}_j \qquad \hat{d}_{ij} = \pi_i \hat{b}_j + (1 - \pi_i)\bar{b}_j$$

Figure 3 - Linear Clustering

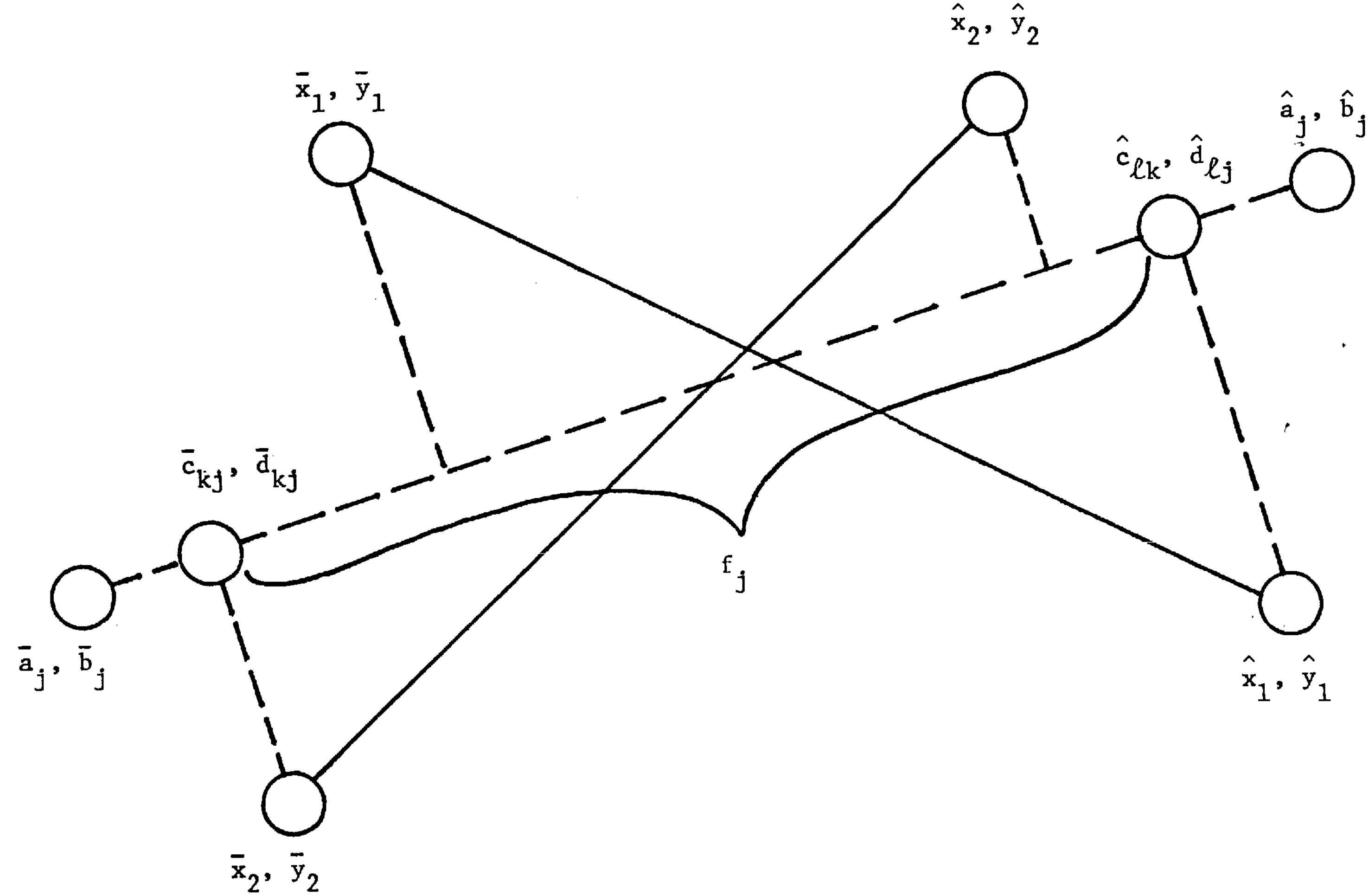


Figure 4 - Linear Clustering

Thus far it has been assumed that $\bar{c}_{ij}$, $\bar{d}_{ij}$ and $\hat{c}_{ij}$, $\hat{d}_{ij}$ exist, but no method of calculating them has been presented. Fortunately, it is possible to calculate these points (or rather their λ_i and π_i values) in closed form, thus avoiding the expense of an iterative procedure. For euclidean and squared euclidean distances, it can be shown that

$$\lambda_i = ((\bar{x}_i - \hat{a}_j)(\bar{a}_j - \hat{a}_j) + (\bar{y}_i - \hat{b}_j)(\bar{b}_j - \hat{b}_j))/$$
$$((\bar{a}_j - \hat{a}_j)^2 + (\bar{b}_j - \hat{b}_j)^2)$$

and

$$\pi_i = ((\hat{x}_i - \bar{a}_j)(\hat{a}_j - \bar{a}_j) + (\hat{y}_i - \bar{b}_j)(\hat{b}_j - \bar{b}_j))/$$
$$((\hat{a}_j - \bar{a}_j)^2 + (\hat{b}_j - \bar{b}_j)^2)$$

For rectilinear distances, it can be shown that

$$\lambda_i = (\bar{x}_i - \hat{a}_j)/(\bar{a}_j - \hat{a}_j) \text{ if } |\bar{a}_j - \hat{a}_j|/|\bar{b}_j - \hat{b}_j| \geq 1$$
$$= (\bar{y}_i - \hat{b}_j)/(\bar{b}_j - \hat{b}_j) \text{ if } |\bar{a}_j - \hat{a}_j|/|\bar{b}_j - \hat{b}_j| < 1$$

and

$$\pi_i = (\hat{x}_i - \bar{a}_j)/(\hat{a}_j - \bar{a}_j) \text{ if } |\hat{a}_j - \bar{a}_j|/|\hat{b}_j - \bar{b}_j| \geq 1$$
$$= (\hat{y}_i - \bar{b}_j)/(\hat{b}_j - \bar{b}_j) \text{ if } |\hat{a}_j - \bar{a}_j|/|\hat{b}_j - \bar{b}_j| < 1$$

A proof of these results is contained in Appendix A.

As with endpoint clustering, the linear clustering problem is also not convex in its objective. Hence, a similar heuristic approach to solving the problem is taken. Given a set of cluster points $\bar{a}_j$, $\bar{b}_j$ and $\hat{a}_j$, $\hat{b}_j$, a transportation problem is solved to yield locally optimal values of the x_{ij}'s.

Given a set of x_{ij}'s, a location problem is solved to yield values of $\bar{a}_j$, $\bar{b}_j$ and $\hat{a}_j$, $\hat{b}_j$. While the transportation problem for linear clustering is identical to that for endpoint clustering, the location problem in linear clustering is more difficult and cannot be solved by use of the Hyperbolic Approximation Procedure. To illustrate this point, consider the formulation of the location problem in linear clustering for euclidean distance shown in Figure 5. This problem has a nonlinear objective function and nonlinear constraints. While one would probably not attempt to solve this problem in this form (since the constraints are all equations, they can be substituted into the objective function), it does illustrate the difficulty in solving this problem: the cross-product terms involving $\alpha_j\bar{c}_{ij}$, $\beta_j\bar{d}_{ij}$, $\alpha_j\hat{c}_{ij}$, and $\beta_j\hat{d}_{ij}$ and

$$\max_{i,k} \{((c_{ij} - c_{kj})^2 + (d_{ij} - d_{kj})^2)^{\frac{1}{2}}\}.$$

Because of the difficulties in solving the location problem in linear clustering, it does not seem possible to solve the problem efficiently. However, suppose that the problem is simplified by eliminating the distance along the cluster line from the objective function for each cluster. The resulting problem is that of minimizing the sum of the distances to the cluster line for all trips in a cluster for each cluster. Morris and Norback [3] have shown that in an optimal solution to this problem, the cluster line must pass through at least two of the trip points (origins or destinations). Thus, the procedure which they suggest for solving this problem is to enumerate all pairs of points as possible locations for the cluster line, and to select that pair for which the sum of the distances to the cluster line is minimal.

Location Problem (Euclidean Distance)

$$\text{Minimize:} \quad \sum_i \sum_j \Big[\lambda x_{ij}((\bar{c}_{ij} - \bar{x}_i)^2 + (\bar{d}_{ij} - \bar{y}_i)^2)^{\frac{1}{2}} +$$

$$\lambda x_{ij}((\hat{c}_{ij} - \hat{x}_i)^2 + (\hat{d}_{ij} - \hat{y}_i)^2)^{\frac{1}{2}} \Big] +$$

$$\sum_j \max_{i,k} ((\hat{c}_{ij} - \bar{c}_{kj})^2 + (\hat{d}_{ij} - \bar{d}_{kj})^2)^{\frac{1}{2}}$$

$$\text{Subject to:} \quad \alpha_j \hat{c}_{ij} + \beta_j \hat{d}_{ij} = \gamma_j \qquad \forall\ ij$$

$$\alpha_j \bar{c}_{ij} + \beta_j \bar{d}_{ij} = \gamma_j \qquad \forall\ ij$$

Where: α_j, β_j, and γ_j are the coefficients in linear form (i.e. $\alpha_j x + \beta_j y - \gamma = 0$) of the cluster line

Figure 5 - Location Problem in Linear Clustering

If one assumes for the location problem in linear clustering that the cluster line must also pass through two trip points, then the problem can be solved by evaluating the objective function for each location of the cluster line and selecting that location with the minimum value. This assumption can be partially justified by noting that the distance along the cluster line is mostly a function of the minimum and maximum x and y coordinates for the trips in the cluster. Thus, it is approximately a constant. If the enumerative solution approach is adopted, then it is possible to consider more realistic values for the e_{ij} terms. Recall that the term e_{ij} represents the contribution of trip i to the total route distance for cluster j. While letting e_{ij} be simply the distance between the origin of trip i and cluster line j plus the distance between the destination of trip i and cluster line j is not a bad approximation, it can be improved upon in many cases. The various combinations of a trip i and a cluster line j which give rise to different ways of evaluating e_{ij} are shown in Figure 6. The trip is indicated by two points connected by a solid line, and the cluster line j is indicated by two points connected by a dashed line.

To illustrate the rationale for the method of evaluating e_{ij} in Figure 6, consider Case 2. Vehicle j must travel out and back to the origin and destination for trip i from the cluster line. Hence, e_{ij} includes distances a and b. Furthermore, because the line which connects the origin for trip i with cluster line j intersects the cluster line to the left of the origin for cluster j, vehicle j must travel the additional distance c in order to pickup trip i. Hence e_{ij} includes the distance c.

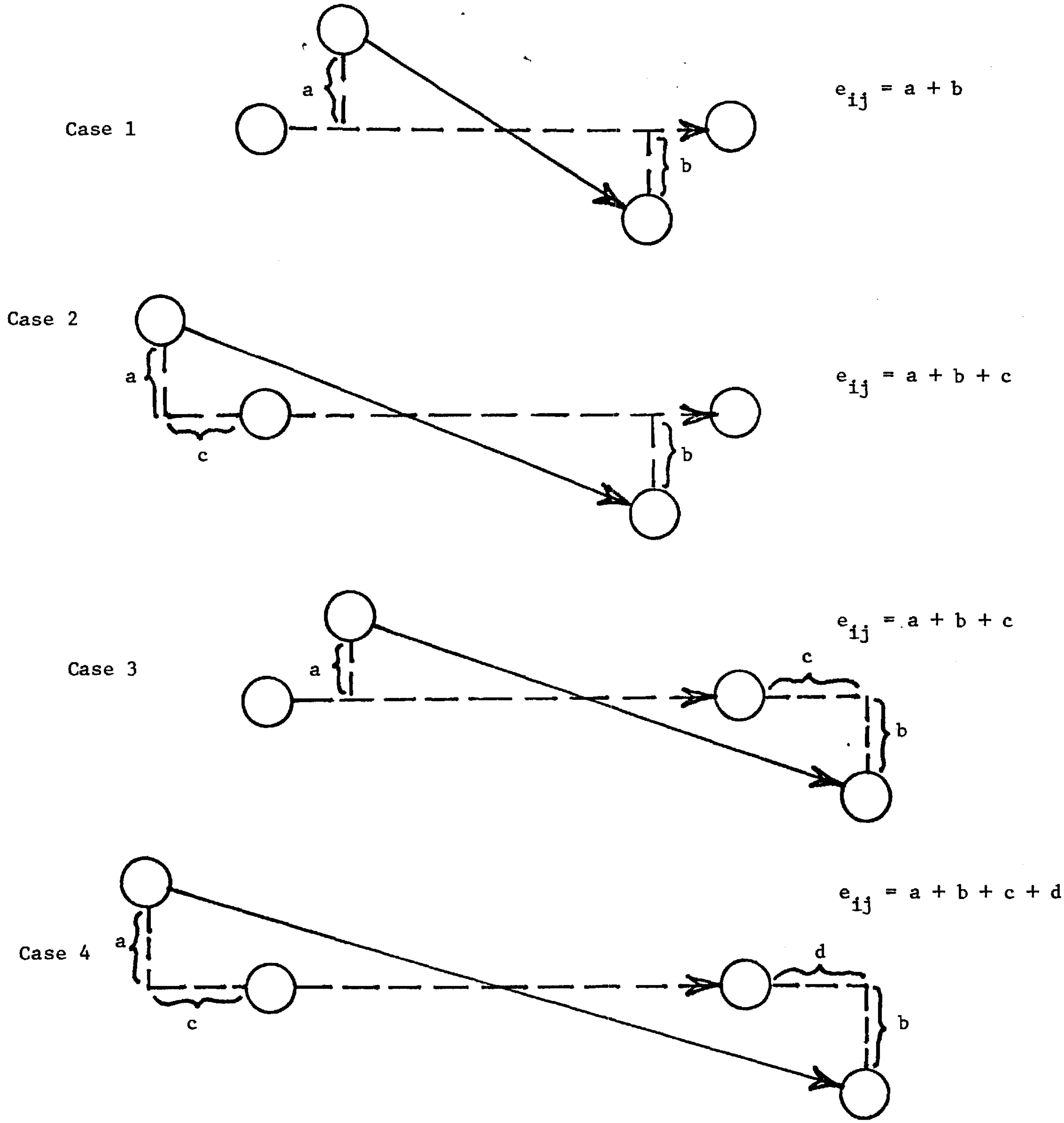
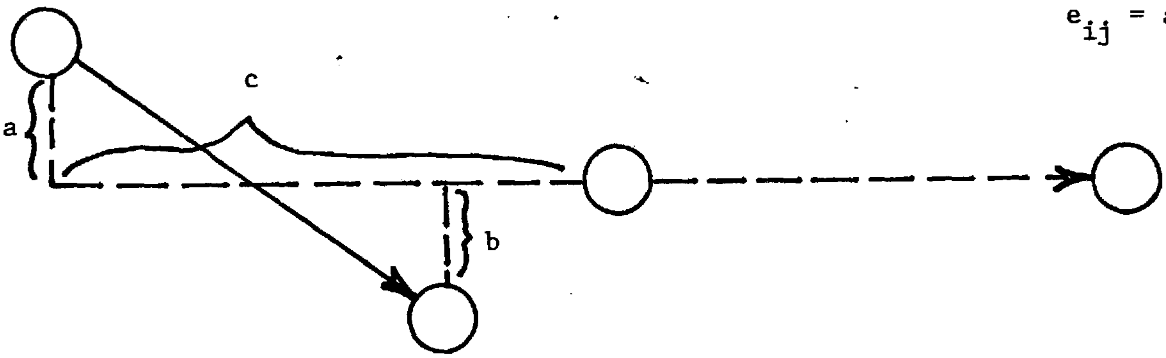


Figure 6 - Evaluation of e_{ij} in Linear Clustering

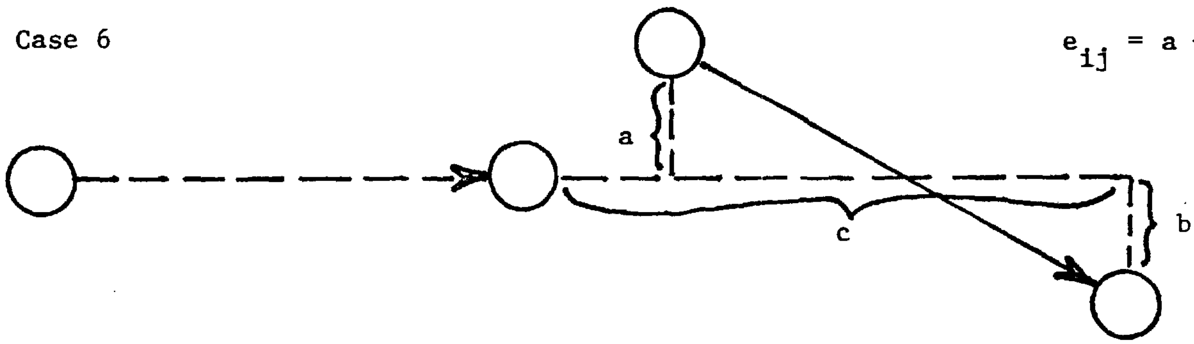
Case 5

$e_{ij} = a + b + c$



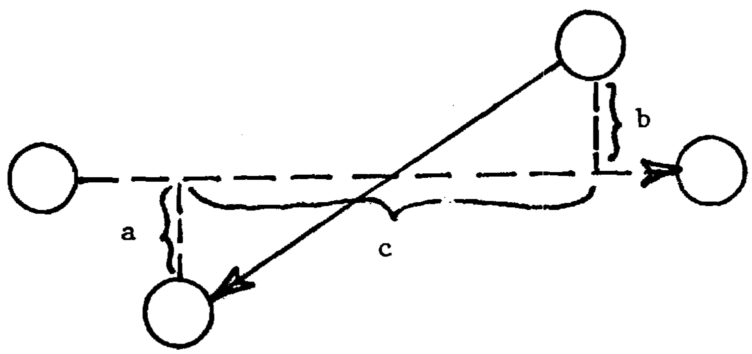
Case 6

$e_{ij} = a + b + c$



Case 7

$e_{ij} = a + b + c$



Case 8

$e_{ij} = a + b + c$

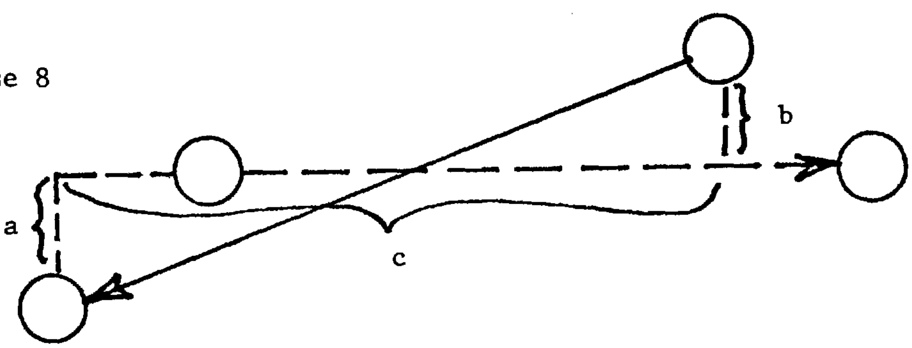
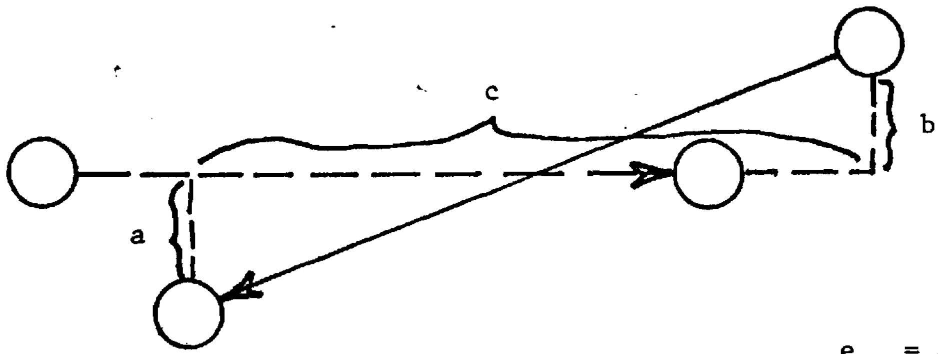


Figure 6 - Continued

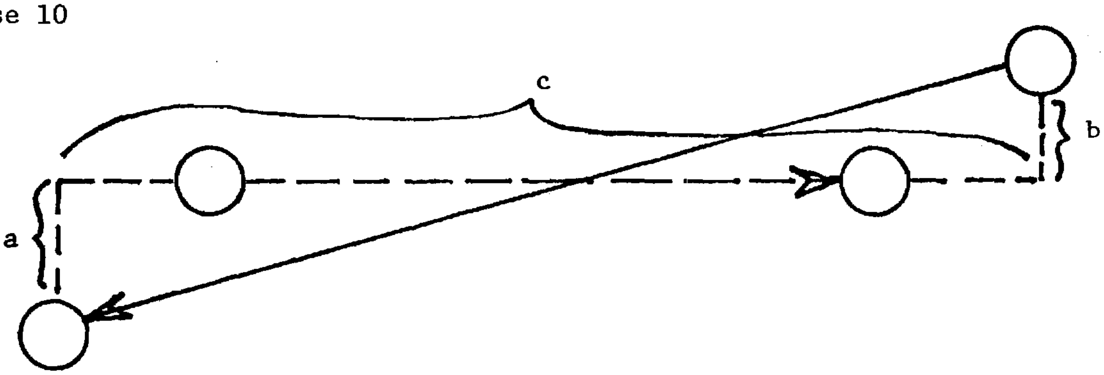
Case 9

$e_{ij} = a + b + c$



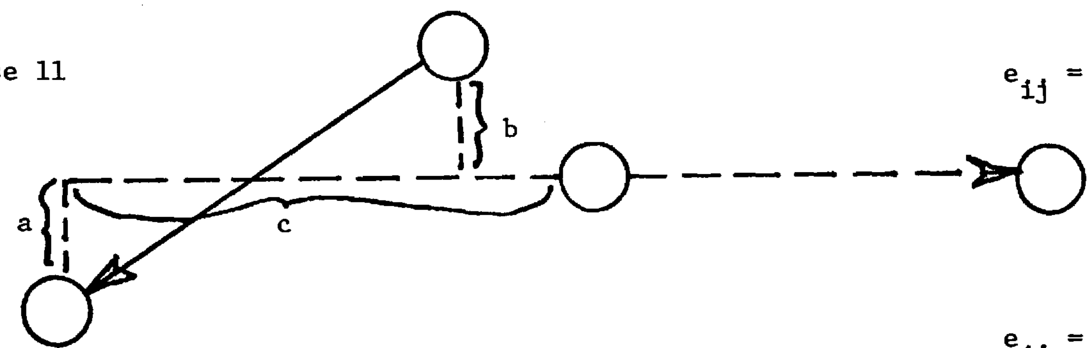
$e_{ij} = a + b + c$

Case 10



Case 11

$e_{ij} = a + b + c$



$e_{ij} = a + b + c$

Case 12

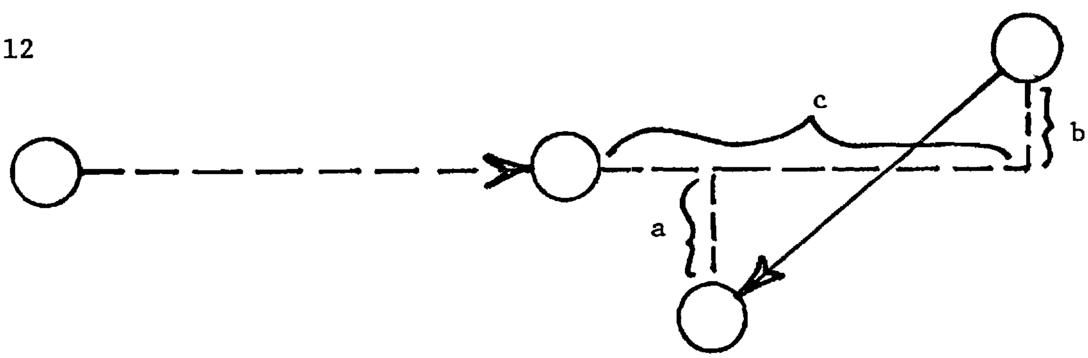


Figure 6 - Continued

5. EXAMPLES OF CLUSTERING

The clustering techniques described above have been implemented on a color graphics terminal. Endpoint and linear clustering models were applied to a sample set of Rochester, New York dial-a-ride data. The results appear quite favorable and are encouraging for future efforts clustering model development.

6. REFERENCES

1. Cullen, F. H., J. J. Jarvis, and H. D. Ratliff, "Set Partitioning Based Heuristics for Interactive Routing and Scheduling," School of Industrial and Systems Engineering Report Series #J-80-1 Georgia Institute of Technology, (1980).

2. Francis, R. and J. White, Facility Layout and Location, Prentice Hall, Englewood Cliffs, New Jersey, 1974.

3. Morris, J. and J. Norback, "A Simple Approach to Linear Facility Location," Transportation Science, Vol. 14, pp. 1-8, 1980.

Appendix A

Verification of Closed Form Expressions

for π and λ in Linear Clustering

First consider euclidean and squared euclidean distances. The problem of finding the point on the cluster line that minimizes the distance between the line and an origin (or destination) point is identical for euclidean and squared euclidean distances. This is true because the minimum of the square root of a function occurs at the same point as that for the minimum of the function.

Let D represent the squared euclidean distance from an origin point to the cluster line. (For simplicity of notation the subscripts are dropped.) Then,

$$D = (\lambda\bar{a} + (1 - \lambda)\hat{a} - \bar{x})^2 + (\lambda\bar{b} + (1 - \lambda)\hat{b} - \bar{y})^2$$

$$\partial D/\partial\lambda = 2(\lambda\bar{a} + (1 - \lambda)\hat{a} - \bar{x})(\hat{a} - \bar{a}) +$$

$$2(\lambda\bar{b} + (1 - \lambda)\hat{b} - \bar{y})(\hat{b} - b)$$

$$= (\lambda(\bar{a} - \hat{a}) + \hat{a} - \bar{x})(\bar{a} - \hat{a}) + (\lambda(\bar{b} - \hat{b}) + \hat{b} - \bar{y})(\bar{b} - \hat{b})$$

$$= \lambda(\bar{a} - \hat{a})^2 + (\hat{a} - \bar{x})(\bar{a} - \hat{a}) + \lambda(\bar{b} - \hat{b})^2 + (\hat{b} - \bar{y})(\bar{b} - \hat{b})$$

$$= \lambda((\bar{a} - \hat{a})^2 + (\bar{b} - \hat{b})^2) - ((\bar{x} - \hat{a})(\bar{a} - \hat{a}) + (\bar{y} - \hat{b})(\bar{b} - \hat{b})) = 0$$

$$\lambda = ((\bar{x} - \hat{a})(\bar{a} - \hat{a}) + (\bar{y} - \hat{b})(\bar{b} - \hat{b}))/((\bar{a} - \hat{a})^2 + (\bar{b} - \hat{b})^2)$$

The argument is similar for π. Hence,

$$\pi = ((\hat{x} - \bar{a})(\hat{a} - \bar{a}) + (\hat{y} - \bar{b})(\hat{b} - \bar{b}))/((\hat{a} - \bar{a})^2 + (\hat{b} - \bar{b})^2)$$

Consider rectilinear distance next. Let D represent the rectilinear distance from an origin point to the cluster line. Then

$$D = |\lambda\bar{a} + (1 - \lambda)\hat{a} - \bar{x}| + |\lambda\bar{b} + (1 - \lambda)\hat{b} - \bar{y}|$$

$$= |\lambda(\bar{a} - \hat{a}) - (\bar{x} - \hat{a})| + |\lambda(\bar{b} - \hat{b}) - (\bar{y} - \hat{b})|$$

This can be rewritten as the following linear programming problem.

Minimize: $D = A + B$

Subject to: $(\bar{a} - \hat{a})\lambda + A = (\bar{x} - \hat{a})$

$(\bar{b} - \hat{b})\lambda + B = (\bar{y} - \hat{b})$

λ, A, B unrestricted

Excluding for the moment the case where $\lambda^* = 0$, we know from the theory of linear programming that if $\lambda^* \neq 0$, then either $A^* = 0$ or $B^* = 0$. If $\lambda^* = 0$, then either $\bar{x} = \hat{a}$ and $A^* = 0$ or $\bar{y} = \hat{b}$ and $B^* = 0$. Hence, when D is minimized either 1)

$$|\lambda(\bar{a} - \hat{a}) - (\bar{x} - \hat{a})| = 0 \Rightarrow \lambda = (\bar{x} - \hat{a})/(\bar{a} - \hat{a})$$

or 2)

$$|\lambda(\bar{b} - \hat{b}) - (\bar{y} - \hat{b})| = 0 \Rightarrow \lambda = (\bar{y} - \hat{b})/(\bar{b} - \hat{b})$$

Thus, at optimality λ can take on only two values; denote them by λ_1 and λ_2. For case 1) to occur then

$$D(\lambda_1) \leq D(\lambda_2)$$

$$|((\bar{b} - \hat{b})(\bar{x} - \hat{a}) - (\bar{a} - \hat{a})(\bar{y} - \hat{b}))/(\bar{a} - \hat{a})| \leq$$

$$|((\bar{y} - \hat{b})(\bar{a} - \hat{a}) - (\bar{b} - \hat{b})(\bar{x} - \hat{a}))/(\bar{b} - \hat{b})|$$

which implies

$$|\bar{a} - \hat{a}| \geq |\bar{b} - \hat{b}|$$

For case 2) to occur then

$$|\bar{a} - \hat{a}| \leq |\bar{b} - \hat{b}|$$

In summary,

$$\lambda = (\bar{x} - \hat{a})/(\bar{a} - \hat{a}) \quad \text{if } |\bar{a} - \hat{a}|/|\bar{b} - \hat{b}| \geq 1$$

$$= (\bar{y} - \hat{b})/(\bar{b} - \hat{b}) \quad \text{if } |\bar{a} - \hat{a}|/|\bar{b} - \hat{b}| < 1$$

The argument for π is similar.

$$\pi = (\hat{x} - \bar{a})/(\hat{a} - \bar{a}) \quad \text{if } |\hat{a} - \bar{a}|/|\hat{b} - \bar{b}| \geq 1$$

$$= (\hat{y} - \bar{b})/(\hat{b} - \bar{b}) \quad \text{if } |\hat{a} - \bar{a}|/|\hat{b} - \bar{b}| < 1$$

APPENDIX A-3

TECHNICAL REPORT:

CHAINING IN THE DIAL-A-RIDE VEHICLE ROUTING PROBLEM

CHAINING IN THE DIAL-A-RIDE VEHICLE ROUTING PROBLEM

BY

David J. Friedman
John J. Jarvis
H. Donald Ratliff

School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332

August 1980

ABSTRACT

A network flow model of chaining in the Dial-A-Ride transportation problem is developed. Since the network flow model employs side constraints, viz certain subsets of arcs may only have one unit of flow between them, it cannot be solved by ordinary network flow techniques.

A Lagrangian relaxation method is developed to handle the side constraints of the network flow problem for chaining. The method appears to work well on small test problems; however, further testing will be required before definite answers about its usefullness can be obtained.

1. INTRODUCTION

Elsewhere [1] a framework for modeling and analysis of demand responsive (dial-a-ride) transportation systems has been discussed. Figure 1 illustrates a dial-a-ride problem in which 11 trips (people) must be accomodated†. Each arc represents an association of origin and destination for the particular trip. The problem is to route the available vehicles through the community so as to satisfy all trip requirements with minimum distance.

In [1] a two stage procedure was developed for analyzing a dial-a-ride system. The trips were first "clustered". The object of clustering is to identify subgroups of trips which will be serviced by the same vehicle by first visiting their origins and then visiting their destinations. Figure 2 depicts five such clusters. By employing different clustering techniques (see [2]) it is possible to generate a number of different clusters, all containing the same trip.

The clusters obtained represent good possibilities for segments (legs) of a vehicle route. The next step in the process is to link ("chain") these clusters into complete vehicles routes. Figure 2 illustrates the chaining concept. Trips 1 and 2 form one cluster, while trips 3, 4 and 5 form another cluster (see Figure 2). In Figure 2, trips 1 and 2 are replaced by a single pseudo (cluster) trip, as is also the case for trips 3, 4 and 5. These cluster trips are then chained together.

The interpretation of chaining is that a single vehicle will service the first set of trips (in Figure 2 these would be trips 1 and 2) and then proceed to service the next set of trips (i.e., trips 3, 4 and 5) in the chain.

---

† Several figures in this report are repeated from the earlier report [1] so that the current report can be read as a self-contained document.

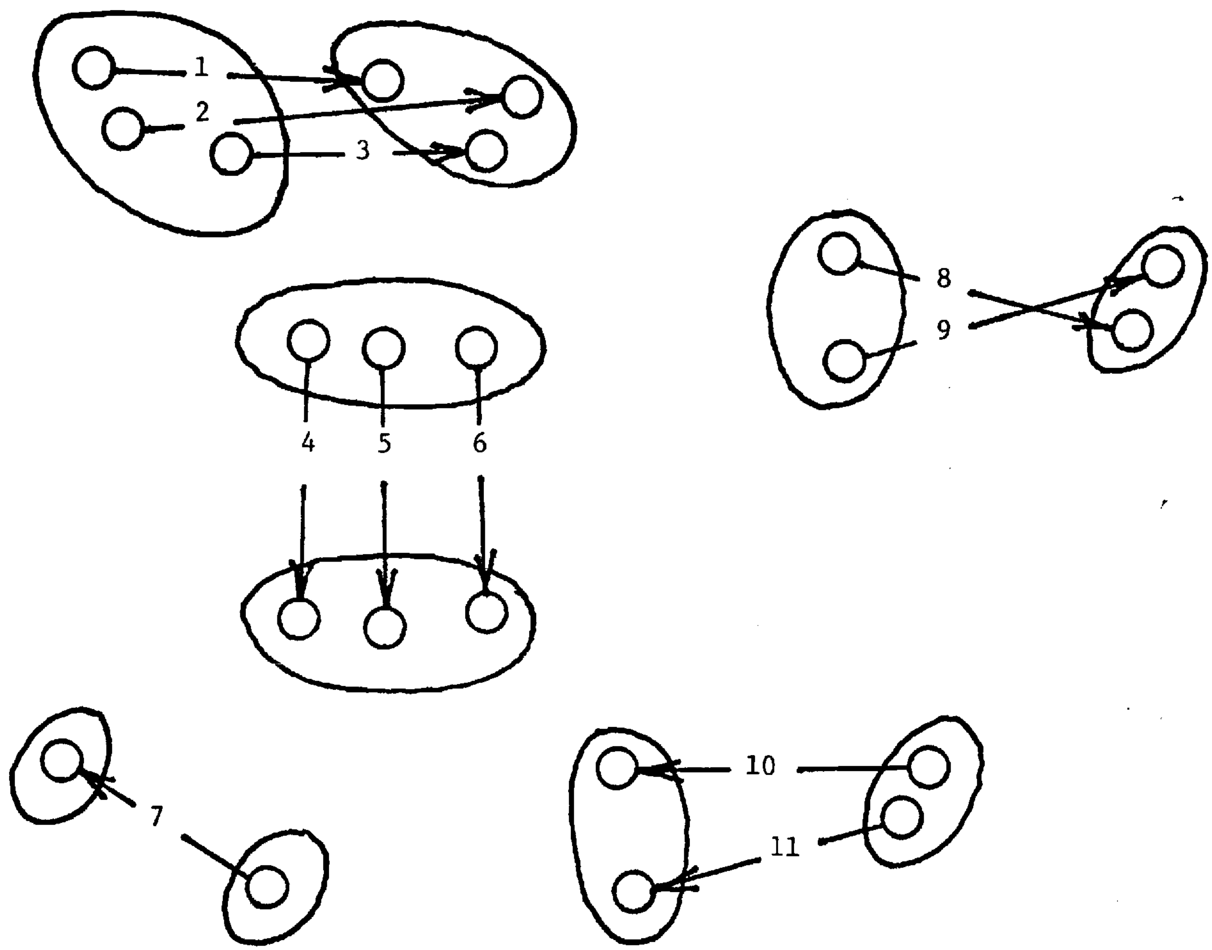
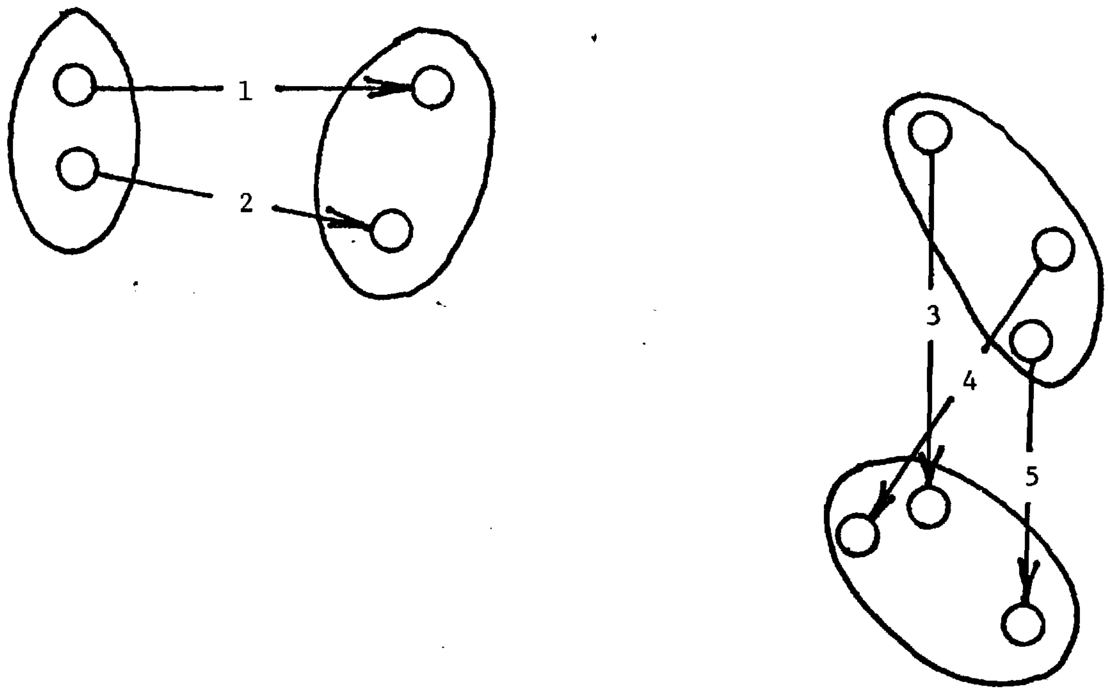
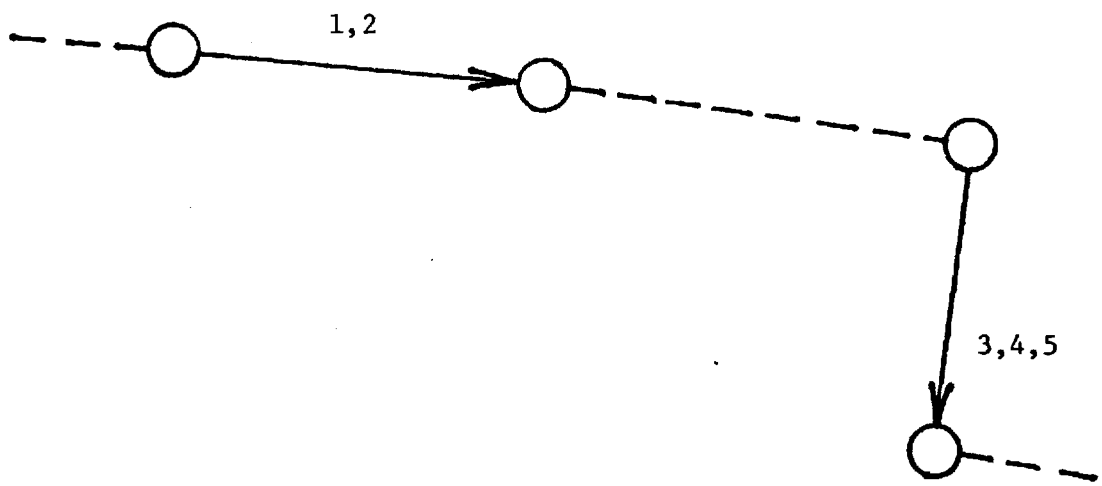


Figure 1. Example of Clusters Resulting from the Clustering Process



a. Two Typical Clusters



b. The Associated Cluster Arcs Chained Together

Figure 2. An Example of Chaining Clusters Together

Figure 3 illustrates the likely vehicle route to service the two clusters.

We shall demonstrate how mathematical models can be developed to aid the human in developing good chains (vehicle routes).

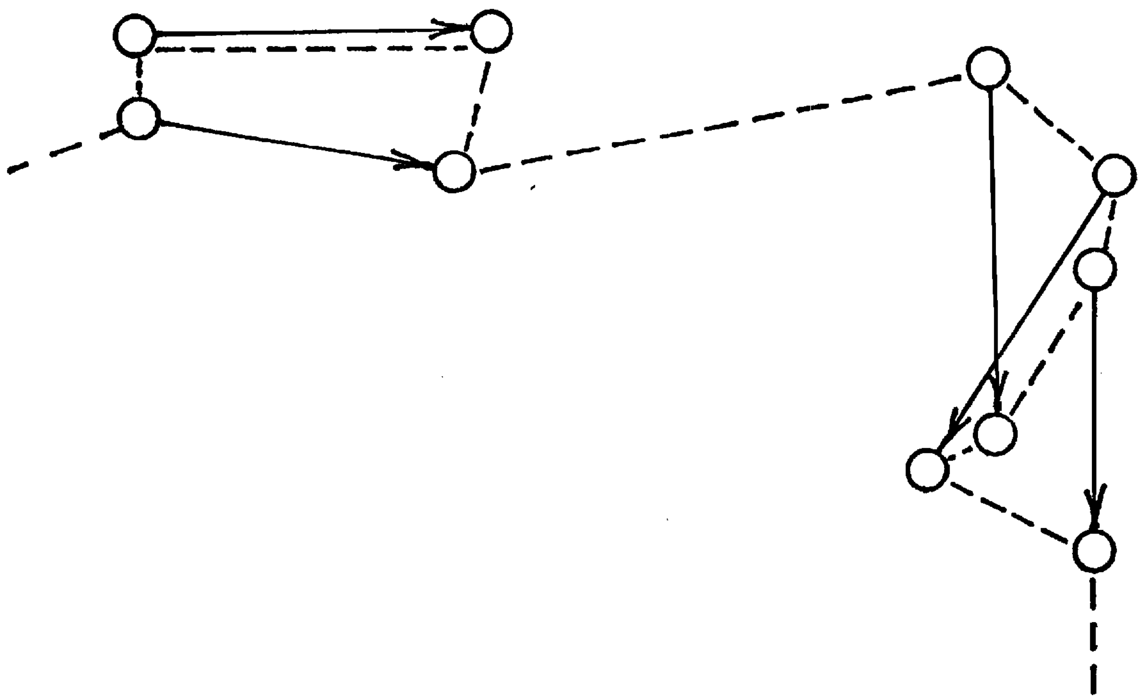


Figure 3. Probable Vehicle Route
for the Chain of Figure 2b

2. NETWORK FLOW/PARTITIONING MODEL FOR CHAINING

The most fundamental model of chaining is based on network flows. In a network we represent each cluster trip by two nodes (an origin and a destination node) and an arc. We then add additional arcs to the network which link the destination node of one cluster arc to the origin node of another cluster arc on the basis of proximity considerations. In Figure 4, we indicate one network flow model for the clusters of Figure 1.

To establish the flow variables for the network flow problem consider the case where each trip appears in exactly one cluster. We assign every arc an upper capacity of 1, the dashed arcs a lower capacity of 0 (zero) and the solid cluster arcs a lower capacity of 1 indicating that these trips must be serviced. Arc flow costs for the network flow problem are the associated vehicle travel distances. If the "starting node" and the "ending node" represents the vehicle storage depot, then we may also assign additional costs to the arcs originating at the starting node to reflect the fixed cost of using each vehicle (provided they are all the same).

If the network flow problem for chaining contains no circuits and if the same trip does not appear in two different clusters in the network, then the resulting chains (vehicles routes) will be valid ones. If either of these two conditions are explicitly included in the model, then the underlying network flow problem becomes considerably more complicated.

We can circumvent some of the problems associated with the difficult problems of chaining by utilizing a partitioning model of chaining. (Actually we substitute one difficulty for another.) In a partitioning model of chaining, we associate a column of the partitioning matrix for each feasible chain (vehicle route) and a row for each trip. Column j contains a 1 in row i if trip i is serviced by chain (vehicle route) j. Otherwise, it contains a 0 (zero). The zero-one variable associated with the columns provide indications

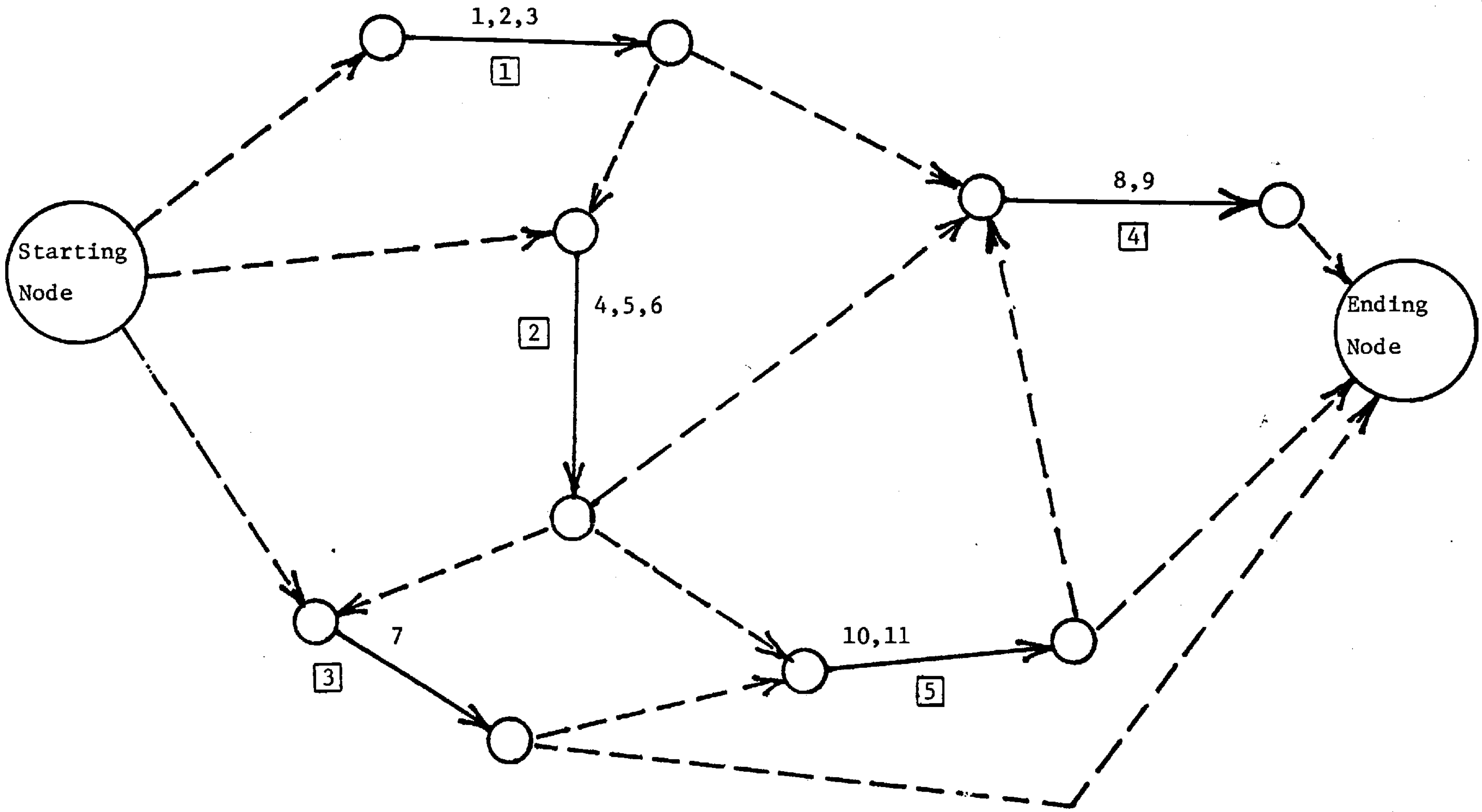


Figure 4. The Network Flow Problem for Chaining of the Clusters in Figure 1

of which chains were selected in the optimal partitioning solution. Table 1 illustrates a partitioning matrix for several chains in Figure 4.

Besides solving the partitioning model optimally (which we would not likely do), the obvious difficulty with such a model is in generating the candidate chains (columns). To aide in this column generation process, we could employ (1) the human (which would certainly be part of any interactive process), (2) a savings approach to combine clusters into chains, (3) a network flow model or (4) some other method. The first two of these column generation approaches should be reasonably intuitive to develop. We shall briefly describe the third one. Suppose we have identified some candidate columns (chains) for the primary concern partitioning problem (e.g. the individual clusters from the clustering process). Then, the partitioning model will yield (1) an optimal solution and (2) a set of row prices. As before, a column j not in the solution appears favorable if

$$\sum_i p_i a_{ij} - c_j > 0.$$

where p_i is the price for row i in the partitioning problem (see [1]). Now, c_j is the cost (distance) of servicing the chain. Also, $\sum_i p_i a_{ij}$ is the cost of servicing the given trips in the current partitioning solution. Consider how these two terms might be represented in the network of Figure 4.

Suppose we associate with each dashed arc the negative of whatever costs that are incurred in traversing the arc (this is the same as before with a sign change). Associated with each clustering arc the quantity $\sum p - c$, where the term $\sum p$ represents the sum of row prices for trips in the cluster and c represents the cost of the cluster. With these costs defined on the network, we seek a path (or paths) in the network, from the starting to the ending node, which has positive total cost. It should be clear that such a

	clusters 1 & 2	clusters 1 & 4	clusters 2 & 5	clusters 1, 2 & 5	clusters 3, 4 & 5	clusters 1, 2 & 4	clusters 1, 2, 3 & 5
	1	2	3	4	5	6	7
1	1	1		1		1	1
2	1	1		1		1	1
3	1	1		1		1	1
4	1		1	1		1	1
5	1		1	1		1	1
6	1		1	1		1	1
7					1		1
8		1			1	1	
9		1			1	1	
10			1	1	1		1
11			1	1	1		1

Table 1 An Example Partitioning Model for Certain Chains in Figure 4

path satisfies the condition for a potentially improving chain, and can be added to the partitioning problem.

Again, with this network flow model (a shortest path model for a single chain) we have the difficulties associated with circuits and with the same trip serviced several times by a single chain. The phenomenon of a trip being serviced several times seems to occur infrequently since in seeking a least cost solution, the model tends to avoid such an instance. When it does occur, an attractive heuristic is to simply delete one of the conflicting clusters from the column being generated. We are currently utilizing the human interactor to handle the case where the flow problem contains circuits. The human breaks the circuits and patches the paths back together. If we restrict the procedure to locating a single improving path each time, the resulting shortest path model lends itself more intuitively to the development of good heuristic procedures for solving both of the major difficulties associated with the flow mode. In particular it is much easier for the human to break the circuits in a single path than in multipaths.

3. THE LAGRANGIAN APPROACH

A possible approach to solving the problem of multiple trips in chaining involves the use of Lagrangian multipliers. When dealing with a general primal problem P(x) of the form:

$$\begin{aligned} P(x):\ \ \min\ \ & cx \\ \text{s.t.}\ \ & Ax \geq b \\ & x \in X \end{aligned}$$

the Lagrangian relaxation, P $\mu(x)$, is defined as

$$\begin{aligned} P_\mu(x):\ \ \min\ \ & cx + \lambda(b-Ax) \\ \text{s.t.}\ \ & x \in X \end{aligned}$$

where λ is a nonnegative vector with one component for each row of A. This relaxation removes constraints from the constraint set and adds them, as a form of penalty, to the objective function. It can be proven that for all $\lambda \geq 0$, $\nu(P_\mu(x)) \leq \nu(P(x))$, where $\nu(\cdot)$ denotes optimal objective value. Thus $P_\mu(x)$ is a valid relaxation of the primal problem. This relaxation will become helpful when considering a network flow problem consisting of chaining clusters together with the added constraints prohibiting multiple trips. Since these added multiple trip constraints rob the problem of its network form, steps should be taken to somehow eliminate these "bad" constraints. It will be attempted in the following manner. First, separate the primal constraint matrix into two submatrices as follows:

$$A = \begin{bmatrix} A' \\ A'' \end{bmatrix}$$

where A' represents the set of all network constraints and A" represents the set of all multiple trip prohibition constraints. Then, relax the primal to $P_\mu(x)$, where

$$P_\mu(x): \quad \min \quad cx + (b''-A''x)$$
$$\text{s.t.} \quad x \epsilon X$$
$$\text{where} \quad X = \{x: A'x \geq b'\}$$

It will be noted that b' is a 1xk' matrix where k' = twice the number of clusters and b" is a 1xk" matrix where k" = the number of trips.

The dimension of k" is intuitive since each trip will have its own constraint prohibiting duplication. It is therefore possible to view λ_i as the penalty for chaining more than one cluster containing the ith trip.

Once the modified problem $P_\mu(x)$ has been constructed, the next step is to solve it using a network flow algorithm, for example the out-of-kilter algorithm. By defining a supersource node, a supersink node and a return arc connecting the sink to the source, a lower bound on the number of chains generated can be established. For instance, if the lower bound on the return arc is 2, at least two chains would be generated (the validity of these chains is yet to be established however). See Figure 5 for an example.

Although the return arc has been forced to 2, three chains are formed ($A \rightarrow B$, $C \rightarrow D$, and $E \rightarrow F \rightarrow G$). Since the final chain is actually a negative flow circuit, the chain was established by simply deleting the largest arc connecting the clusters. The chaining algorithm thus proposed will iteratively solve an out-of-kilter problem and at each iteration store all valid

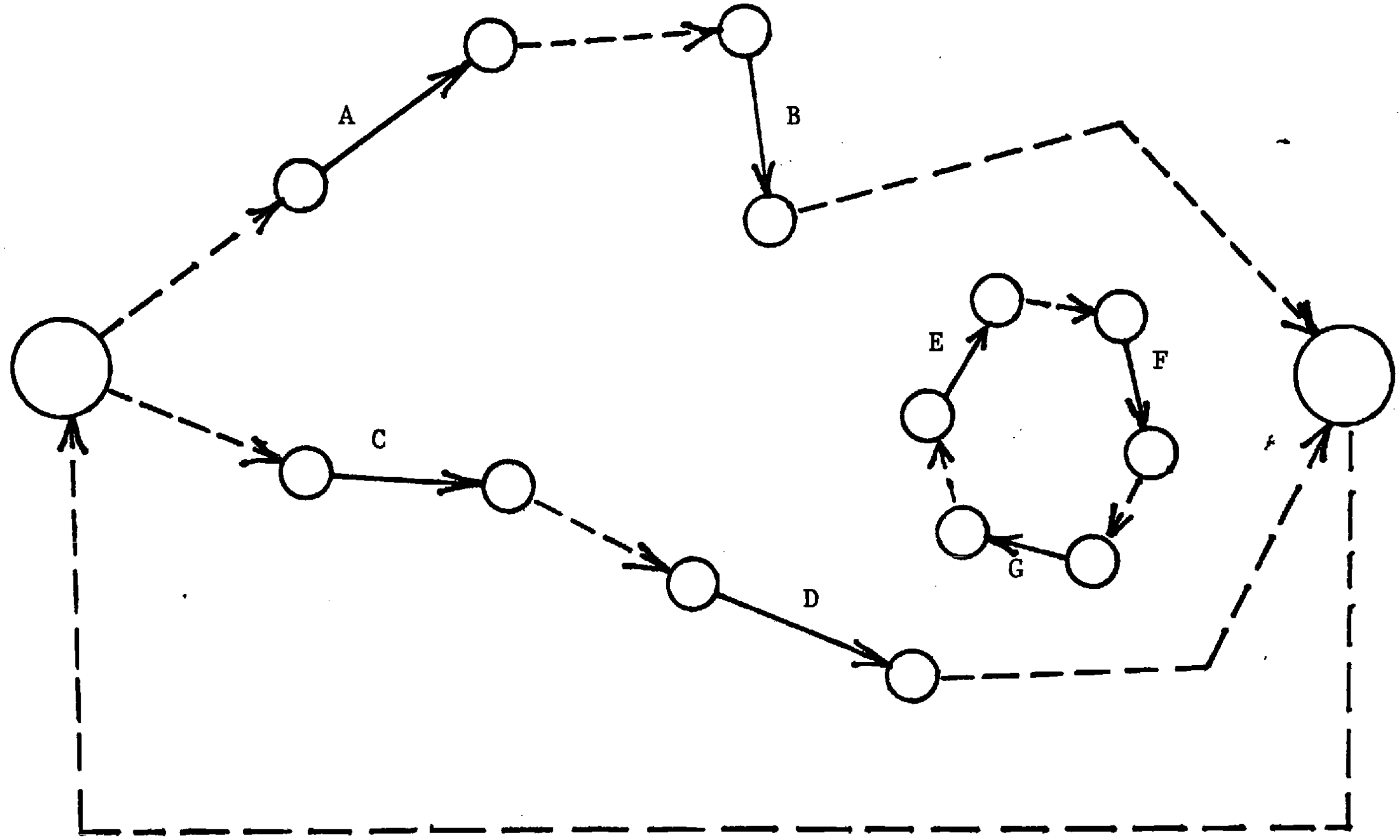


Figure 5. Example of A Solution Generated in the Chaining Network

chains as columns for the set covering model. By increasing the λ_i for the violated trips on invalid chains, new solutions to the algorithm will be generated during the next iteration. The idea behind increasing the λ_i, of course, is to make clusters violating the one trip constraint less desirable. The λ_i will be increased until all clusters containd trip i (except possibly one) are driven from the chain.

Consider an example of this method. Figure 6 represents twelve clusters that are available upon termination of the clustering routine. The trips within each cluster are shown in Table 2 and the costs of traveling from one cluster to another (along with the benefits of servicing each cluster) is shown in Table 3.

Initially, the problem was solved using $\lambda_i = 0$ for every i and a return arc of 1, with the result shown in Figure 7. As can be seen, a chain of $x_{12} \rightarrow x_8 \rightarrow x_6 \rightarrow x_2 \rightarrow x_1$ was formed. This is not, however, a valid chain since trip 6 is contained in both cluster 6 and cluster 2. If a return arc constraint of two is used (Figure 8) two paths are generated, neither of which constitutes a valid chain. It will be noted, however, that the raising of the lower bound decreased the value of the objective function, thus increasing the benefits. If this were not the case, it would indicate that the chains being added (or augmented) were not particularly good ones. Taking the simpler case of only one conflict (with the return arc constraint of 1), we increase the value of λ_6 to 5 (since 6 is the violated trip) and resolve the conflict. Figure 9 indicates that two chains are formed, with one a negative flow circuit. Since neither path contains any conflicts, two valid chains have been generated. Their corresponding set cover columns are: $[0,0,0,0,0,0,0,1,0,0,1,1,0,1,0,1]^t$ for the negative flow circuit of $x_8 \rightarrow x_9 \rightarrow x_{12}$ and $[1,1,1,1,0,1,1,0,0,0,0,0,0,0,0,0]^t$ for $x_2 \rightarrow x_1$. Computing

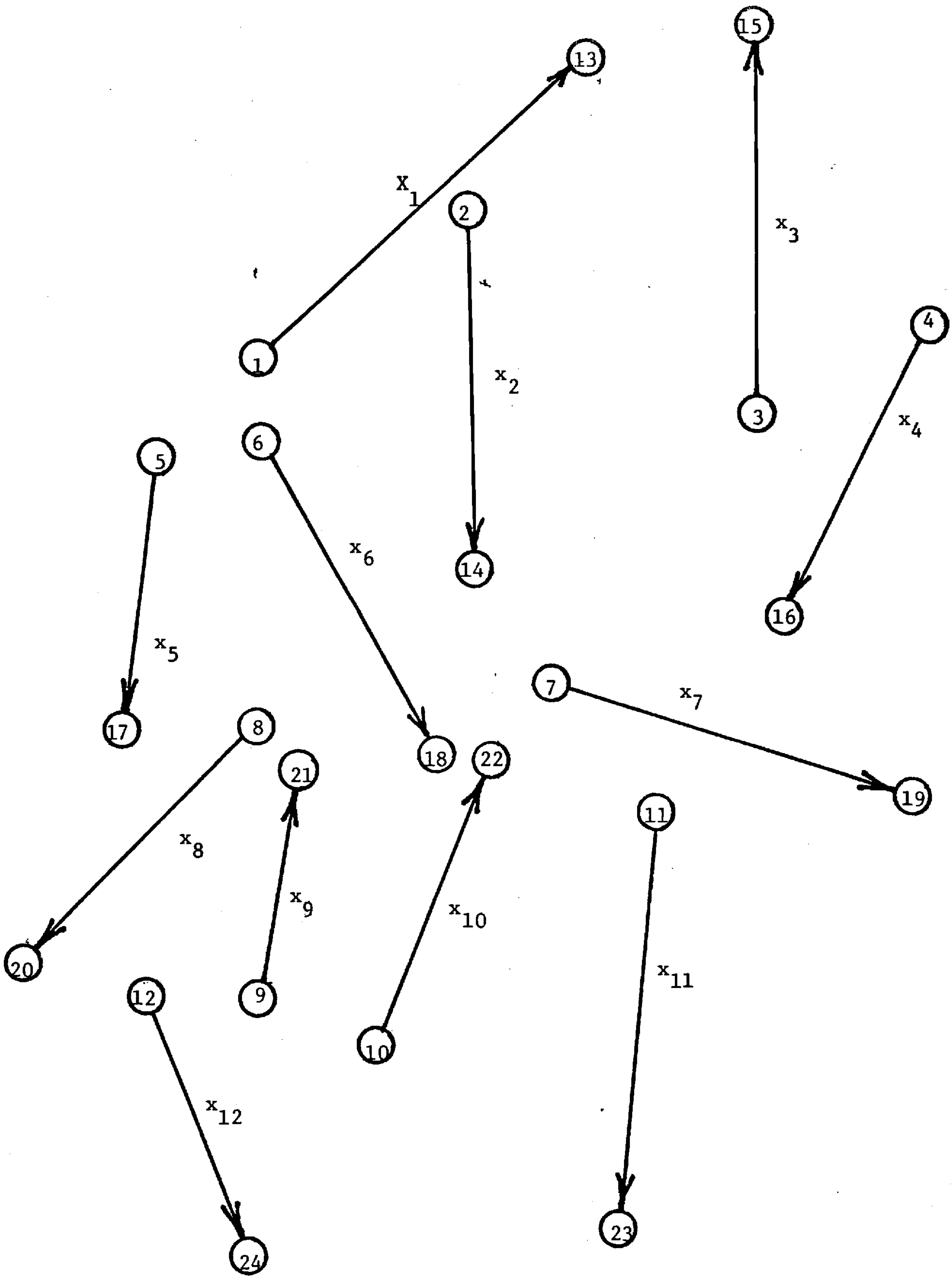


Figure 6. Twelve Clusters Available for Chaining

TRIP	CLUSTER
1	1
2	2
3	1,3
4	1,3
5	4
6	2,5,6
7	2,4
8	5,8
9	6,7
10	7,11
11	8
12	9,10
13	10
14	12
15	11
16	12

CLUSTER	TRIP
1	1,3,4
2	2,6,7
3	3,4
4	5,7
5	6,8
6	6,9
7	9,10
8	8,11
9	12
10	12,13
11	10,15
12	14,16

Table 2 Correspondence Between Trips and Clusters

BEGIN NODE	END NODE	COST
1	13	-13
14	1	7
15	1	12
18	1	17
2	14	-16
13	2	10
15	2	11
18	2	7
3	15	-9
14	3	11
18	3	21
19	3	23
4	16	-7
15	4	7
19	4	8
23	4	9
5	17	-9
20	5	4
18	5	10
21	5	10
6	18	-8
19	6	5
20	6	6
23	6	8
7	19	-5
15	7	15
16	7	15
23	7	8
8	20	-9
21	8	8
22	8	11
24	8	5
9	21	-2
20	9	3
22	9	9
24	9	9
10	22	-7
19	10	4
21	10	10
23	10	7
11	23	-6
21	11	14
22	11	10
24	11	17
12	24	-7
20	12	18
21	12	9
23	12	9

Table 3 Travel Cost Between Clusters

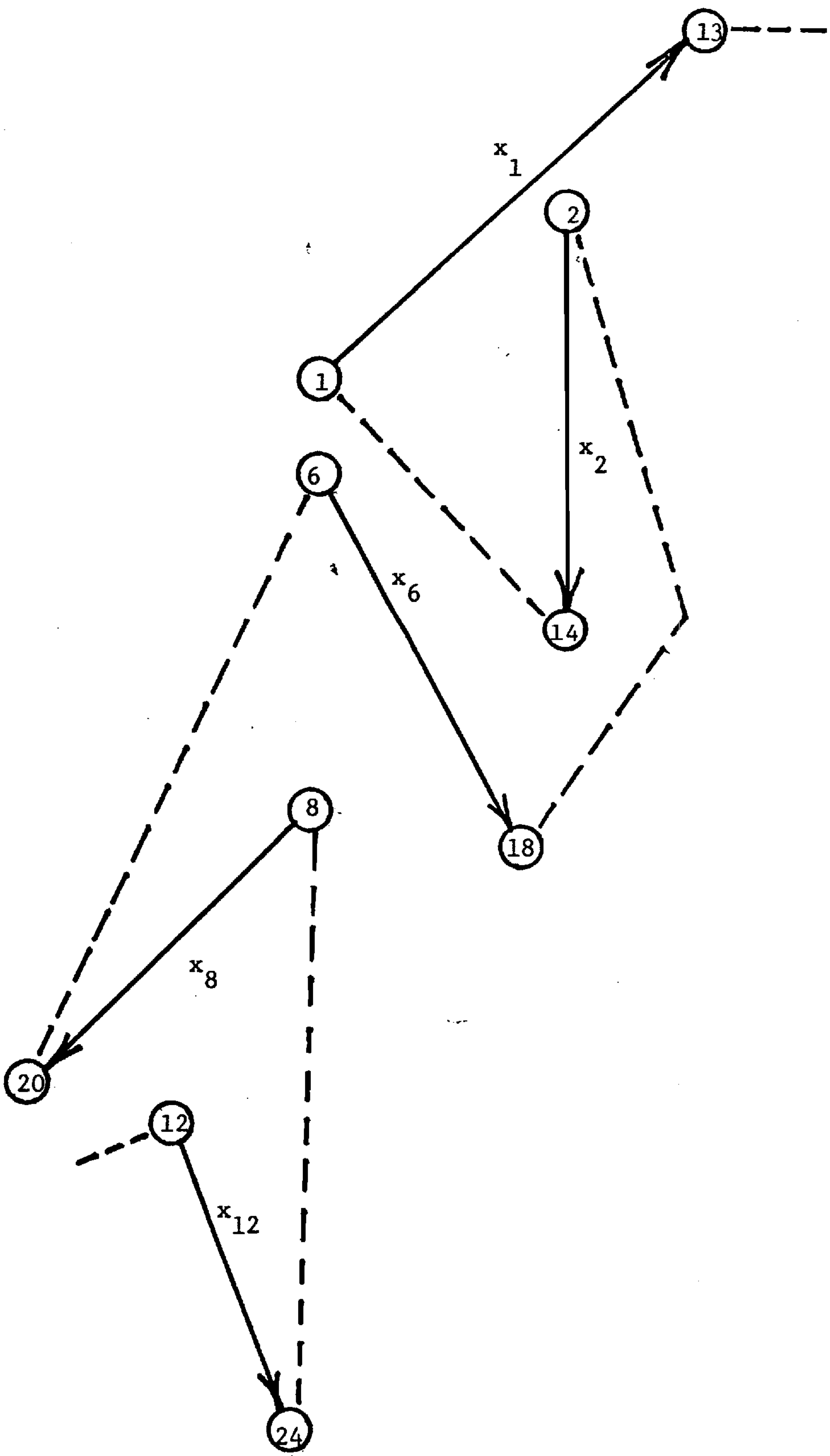


Figure 7. Example of a Chain Through the Clusters

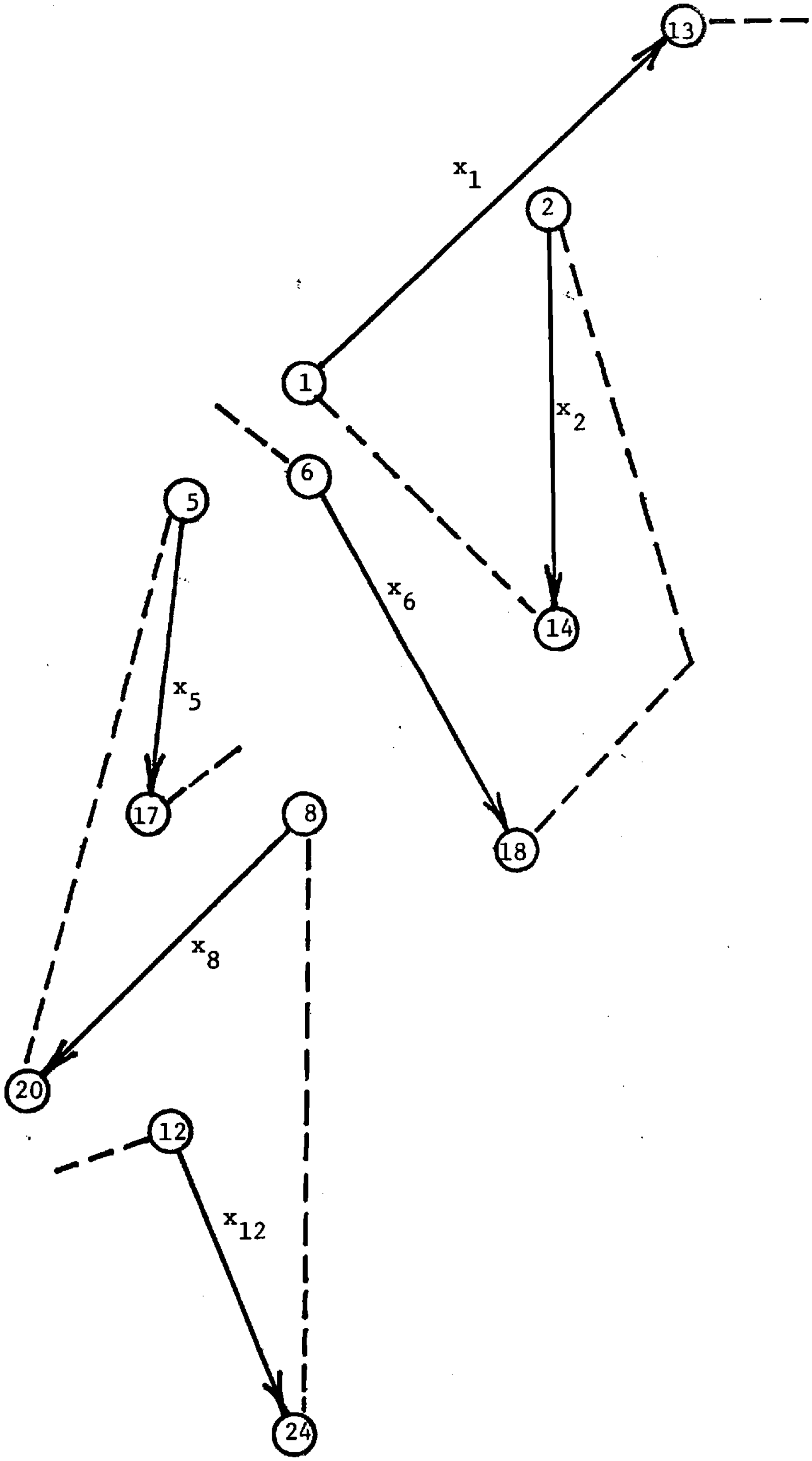


Figure 8. The Network Flow Solution for a Two Chain Requirement

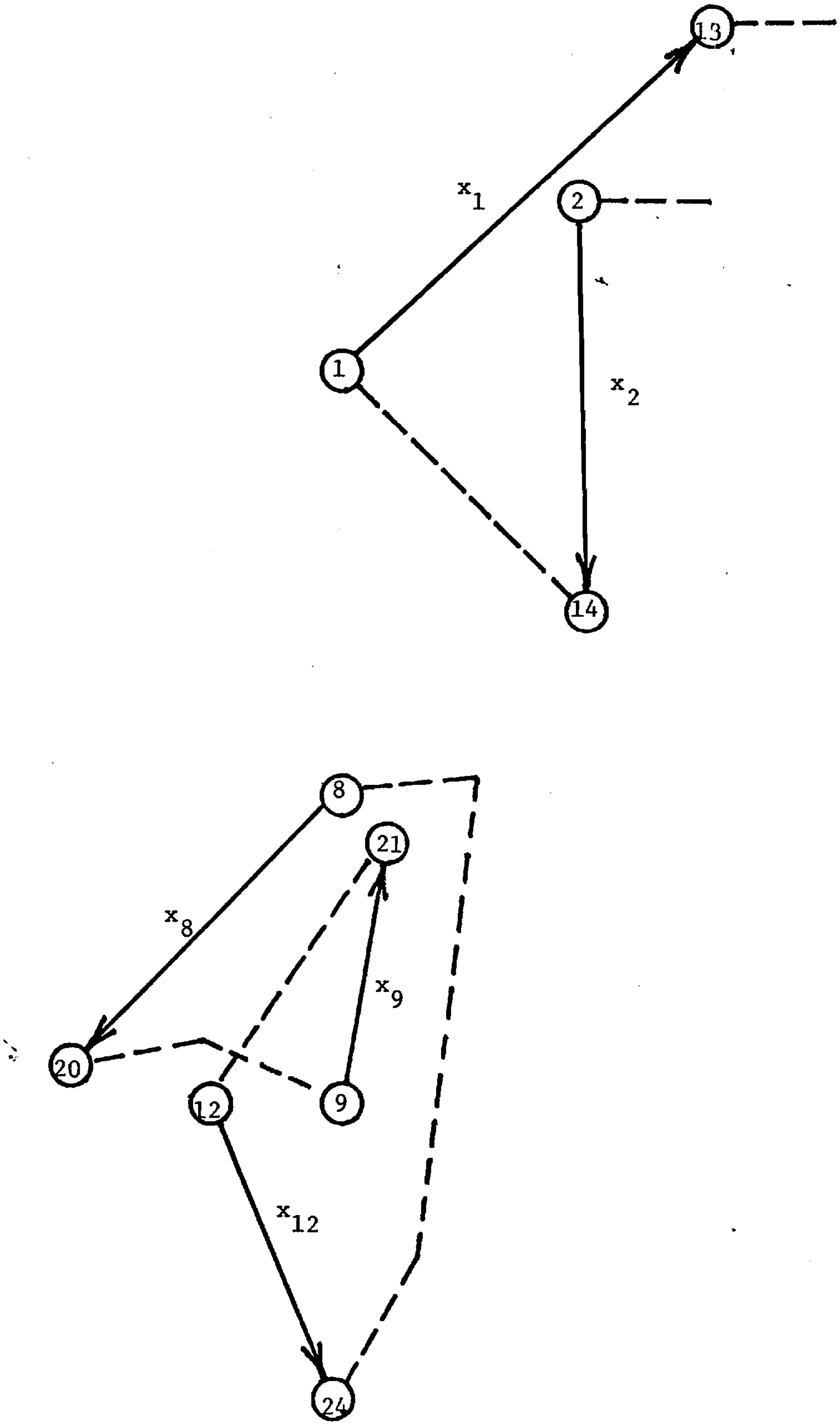


Figure 9. The Network Flow Solution for $\lambda_6 = 5$

the reduced costs for $x_2 \rightarrow x_1$ the path is source → node 2 → node 14 → node 1 → node 13 → sink and its reduced cost is 0 + (-16) + 7 + (-13) + 0 = -22. Similarly, for $x_8 \rightarrow x_9 \rightarrow x_{12}$ the reduced cost is (-9) + 3 + (-2) + 9 + (-7) + 5 = -1. Now the most expensive arc of cost 9 is deleted (eliminating the circuit). Thus, the true reduced cost is -1 -9 = -10.

An alternate way to approach this problem is to increase only those costs corresponding to clusters containing violated trips. Thus, in Figure 7, where clusters 6 and 2 both contained trip 6, only those two clusters would be increased by 5 (in the Lagrangean relaxation approach all clusters containing trip 5 would be penalized, whether they are in the basis or not). The difference will be noted in Figure 10. Now there is a conflict in path 2 which contains clusters 12, 8 and 5. If the costs for cluster 5 had been penalized for having trip 6 (as was done in Figure 8), the cluster would not have entered into the basis, thus avoiding the conflict. More generally, penalizing only the violated clusters goes against the very spirit of the method developed, since it only addresses selected members of the A" matrix.

Finally, the lower bound is raised from 1 to 2 in the final model (with $\lambda_6 = 5$). This breaks up the negative flow circuit (Figure 11) and generates a new column $x_{12} \rightarrow x_8$ which is a proper subset of the column generated by the negative flow circuit. Since its reduced cost is (-7) + 5 + (-9) = -11 and it is not dominated by the negative flow cycle column, it is a valid column.

Summarizing the example, there were three valid columns added to the partitioning model for the clusters presented. By varying other parameters (such as lower bounds of each cluster, which could be made 1 to force inclusion in the basis), a number of other possibly valid columns could be generated.

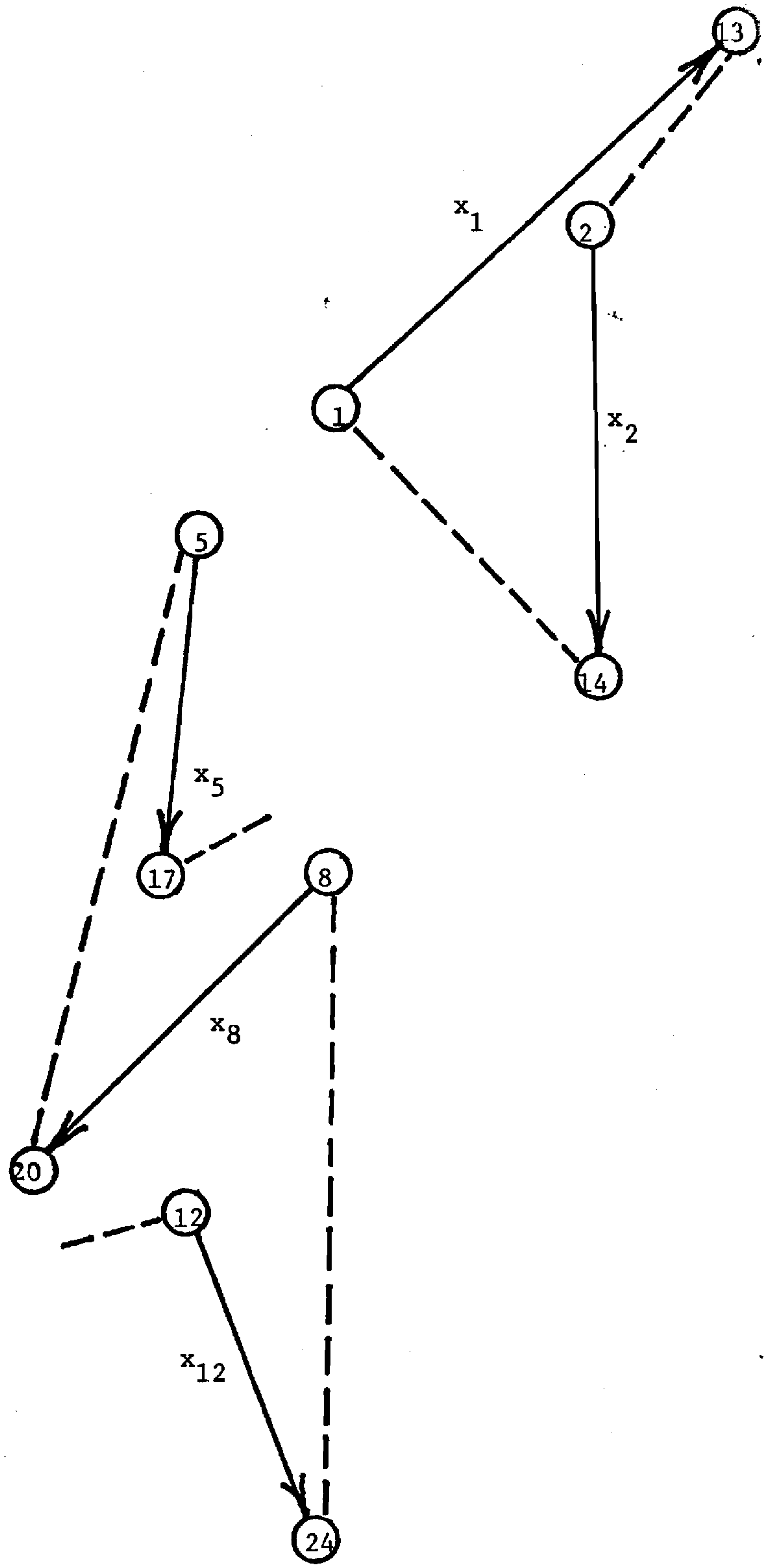


Figure 10. The Network Flow Solution When Only Certain Clusters Containing Trip 6 Are Penalized by 5.

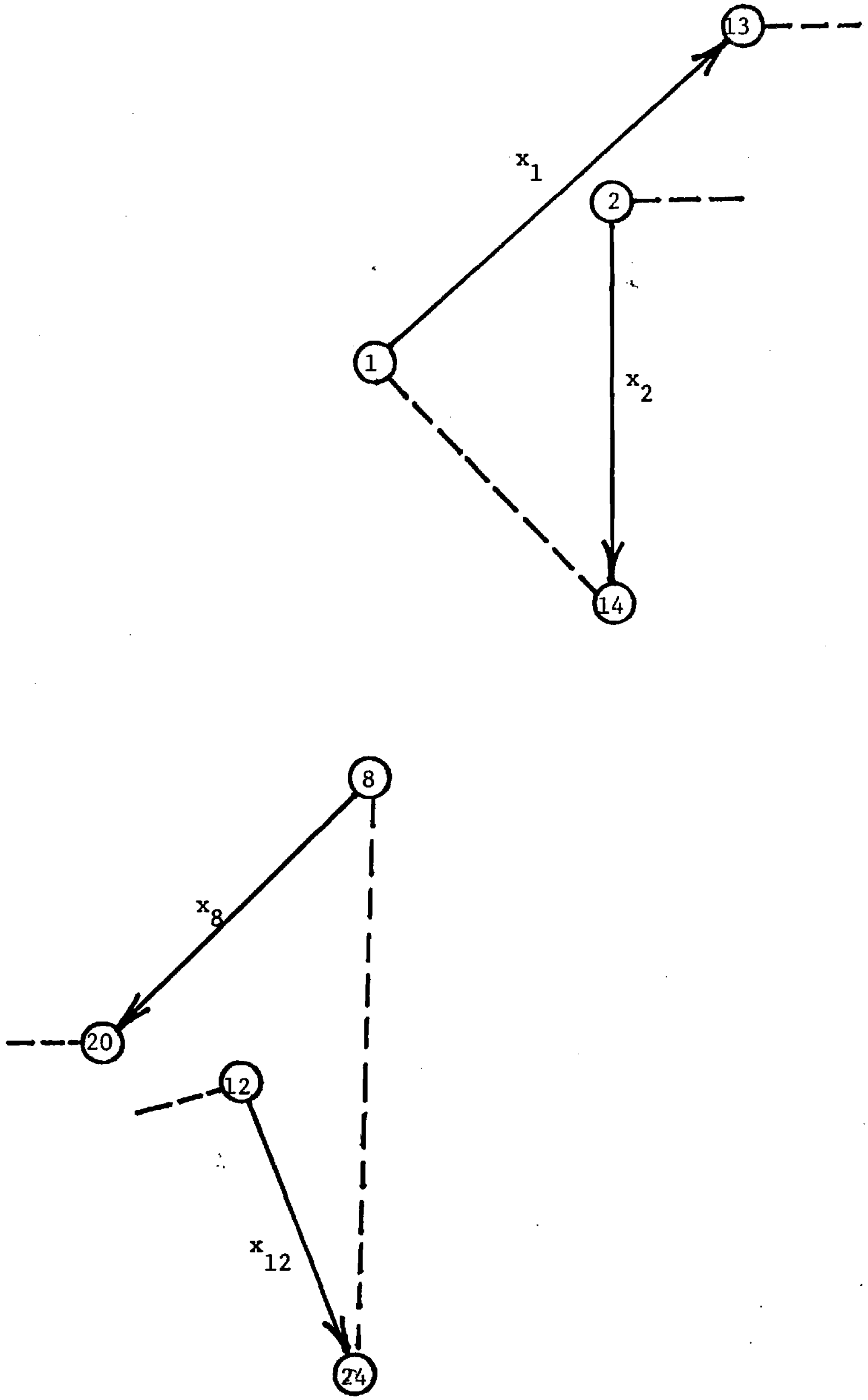


Figure 11. The Network Flow Solution for A Two Chain Requirement With $\lambda_6 = 5$

It will be noted that it was not explained by how much λ_i should be incremented so as to drive the violated clusters out of the basis. This problem will be addressed later on in the extensions section. For the time being, let it just be noted that for some nonnegative value of λ a "better" solution may be determined.

4. OTHER MODELS FOR CHAINING

It has also been suggested that a shortest path algorithm might be used to address the problem of chaining. The mathematical formulation of the shortest path problem is

$$\min \quad cx$$

$$\text{s.t.} \quad \sum_{j=1}^{m} x_{ij} - \sum_{k=1}^{m} x_{ki} = \begin{cases} 1 & \text{if } i = 1 \\ 0 & \text{if } i \neq 1 \text{ or } m \\ -1 & \text{if } i = m \end{cases}$$

$$x_{ij} = 0, 1 \quad i,j = 1,2,\ldots,m$$

As can be seen, this is a special case of the out-of-kilter algorithm, with the return arc restricted to one. Since the general case is the more useful one here, the intuitive approach of using the shortest path algorithm will not be considered further.

5. EXTENSIONS

The first extension which will be discussed is calculation of "good" values for the Lagrangian multipliers. A good value, in this context, constitutes a value which will drive at least one violated cluster from the basis. To do this, the dual variables arrived at in the termination of the out-of-kilter algorithm were examined. Considering the example previously analyzed, after the first iteration (with all $\lambda_i = 0$) there was one conflict, which arose because clusters 2 and 6, both elements of the basis, both contain trip 6. The dual variables are shown in Table 4. The reduced cost for each cluster ppears in Table 5 for six values of λ_6. It will be noticed that if the reduced cost is less than zero, the arc is at its lower bound; similarly, if the reduced cost is greater than zero, the arc is at its upper bound. If the reduced cost is zero, however, the out-of-kilter algorithm states that an arc could be at its lower bound or upper bound thus (as evidence by the data in Table 5) there is no obvious way to find the minimum λ needed to change the basis. There is, however, a strong chance that further research in this area will yield a method for calculating the Lagrangian multipliers which is superior to the haphazard method previously employed.

Another possible extension of the Lagrangian multiplier method outlined is investigation into the possibility that a certain circumstance or set of circumstances might cause the upper bound on the return arc to be varied. Since this upper bound constraint does effect the chains produced (as shown in the previous example), it is possible that a certain upper bound or sequence of upper bounds, should be used to generate chains.

A second, more general, extension that should be pursued is the incorporation of time constraints into the model. This would most likely be done through the use of time slices on, for example, an hour-by-hour basis. The

NODE	PI
1	11
13	0
14	4
15	1
18	3
2	10
3	11
19	6
4	8
16	1
23	3
5	9
17	0
20	5
21	3
6	11
7	11
8	11
22	1
24	6
9	8
10	10
11	11
12	11
25	11
26	0

Table 4 Dual Variables for the OOK Solution

		λ_6					
		0	1	2	3	4	5
	1	2	3	3	4	4	3
	2	10	8	7	6	5	4
	3	-1	-1	-1	0	0	-1
C	4	0	0	0	0	0	0
L	5	0	0	0	0	0	0
U	6	0	0	0	-1	-1	0
S	7	0	0	0	0	0	0
T	8	3	3	2	2	1	0
E	9	-3	-3	-2	-2	-1	0
R	10	-2	-2	-2	-1	-1	-2
	11	-2	-2	-2	-2	-2	-2
	12	2	2	2	2	2	1

Smallest value of λ_6 which eliminates multiple trip within one cluster. $\therefore$ $\lambda_6^* = 3$

◯ indicates elements of basis, i.e. flow is one.

Table 5. Reduced Costs for Various Values of λ_6

clustering would then be done on an hour-by-hour basis so the chaining would be forced to take into account these time precedent relationships. While this has not been addressed in the chaining study thus far, it remains a worthwhile topic for future study.

Another possibility for chaining is to utilize a savings approach similar to that described in [1]. The development of such an approach would follow directly from the method described there.

6. CONCLUSIONS

Futher development and testing of several methods will be required before definitive conclusions can be drawn about chaining, in both theory and practice.

7. REFERENCES

1. Cullen, F. H., J. J. Jarvis, and H. D. Ratliff, "Set Partitioning Based Heuristics for Interactive Routing," School of Industrial and Systems Engineering Report Series #J-80-1, Georgia Institute of Technology (1980).

2. Jarvis, J. J., H. D. Ratliff, and R. T. Lewis, "Clustering in the Dial-A-Ride Vehicle Routing Problem," School of Industrial and Systems Engineering Report Series #J-80-15, Georgia Institute of Technology (1980).

APPENDIX B

COMPUTER PROGRAMS

B-1

Chromatics (BASIC)
Data Generating Program for
Dial-A-Ride

```
100 N$="MENU2"
105 PRINT CHR$(12);
110 GOSUB 2000
120 GOSUB 2300
130 DIM PMENU(MX,5)
140 FOR I=1 TO MX
150 FOR J=1 TO 5
160 PMENU(I,J)=MENU(I,J)
170 NEXT J
180 NEXT I
185 PX=MX
190 ERASE MENU,M$,HDMENU
200 N$="MENU1"
210 GOSUB 2000
220 GOSUB 2300
230 PRINT "~G~C7+0,0,511511";
240 PRINT "*0,400511400350400350511";
250 DIM NW(50,5)
890 OUT&H90,0
900 ON ERROR#2 GOTO 1000
910 GOTO 910
1000 REM *** LIGHT PEN INTERRUPT ROUTINE
1010 X=CURSX(4)
1020 Y=CURSY(4)
1030 IF Y < 400 THEN GOTO 5000
1040 IF X > 350 THEN GOTO 3000
1050 GOSUB 2700
1070 OUT&H90,0
1080 RESUME
2000 REM *** COLOR MENU READ-IN ROUTINE
2010 DOS"OPEN 5 R "+N$
2020 INPUT #5; MX,SMX
2030 DIM MENU(MX,5),M$(SMX),HDMENU(SMX,3)
2040 FOR I=1 TO MX
2050 INPUT #5; MENU(I,1),MENU(I,2),MENU(I,3),MENU(I,4),MENU(I,5)
2060 NEXT I
2061 IF SMX = 0 THEN GOTO 2070
2062 FOR I = 1 TO SMX
2064 INPUT #5; HDMENU(I,1),HDMENU(I,2),HDMENU(I,3),M$(I)
2066 NEXT I
2070 DOS"CLOSE 5"
2080 RETURN
2300 REM *** MENU DRAW ROUTINE
2310 PRINT "~G~K";
2320 FOR I=1 TO MX
2330 PRINT "~F~C";
2340 PRINT USING "#";MENU(I,5);
2350 PRINT "+";
2360 PLOT MENU(I,1),MENU(I,2),MENU(I,3),MENU(I,4)
2362 PRINT "~C7~L+";
2364 PLOT MENU(I,1),MENU(I,2),MENU(I,3),MENU(I,4)
2370 NEXT I
2390 PRINT CHR$(21);
2395 IF SMX=0 THEN RETURN
2400 FOR I = 1 TO SMX
2410 PRINT "~G~U";
2420 PLOT HDMENU(I,1),HDMENU(I,2)
2430 PRINT "~C";
2440 PRINT USING "#"; HDMENU(I,3);
2450 PRINT CHR$(21);M$(I);
2460 NEXT I
2470 RETURN
2500 REM *** MENU LOCATE ROUTINE
2502 PRINT CHR$(27);"OA1";"~W351401510510~;8";CHR$(12);"~;F";CHR$(27);"OA0";
```

```
2510 FOR I=1 TO MY
2520 IF X < MENU(I,1) OR X > MENU(I,3) THEN GOTO 2550
2530 IF Y < MENU(I,2) OR Y > MENU(I,4) THEN GOTO 2550
2540 GOTO 2580
2550 NEXT I
2560 I=0
2570 RETURN
2580 PRINT CHR$(7);
2582 PRINT "~C7~;8~1+";
2584 PLOT MENU(I,1),MENU(I,2),MENU(I,3),MENU(I,4)
2586 PRINT "~;F~L~2";
2590 RETURN
2700 REM *** PRIMARY MENU LOCATE ROUTINE
2702 PRINT CHR$(27);"CA1";"~K0014013495100";"E";CHR$(12);"~;F";CHR$(27);"CA0";
2710 FOR I= 1 TO PX
2720 IF X < PMENU(I,1) OR X > PMENU(I,3) THEN GOTO 2750
2730 IF Y < PMENU(I,2) OR Y > PMENU(I,4) THEN GOTO 2750
2740 GOTO 2780
2750 NEXT I
2760 I=0
2770 RETURN
2780 PRINT CHR$(7);
2782 PRINT "~K7~;8~1+";
2784 PLOT PMENU(I,1),PMENU(I,2),PMENU(I,3),PMENU(I,4)
2786 PRINT "~;F~L~2";
2790 F=I
2800 B1=0:B2=0:E1=0:E2=0
2810 IF F=3 THEN GOSUB 8000
2820 IF F=4 THEN GOSUB 9000
2825 IF F=6 THEN GOSUB 9500
2830 RETURN
3000 REM *** COLOR CHANGE ROUTINE
3010 GOSUB 2500
3020 IF I = 0 THEN GOTO 1070
3030 CR=MENU(I,5)
3050 GOTO 1070
4000 REM ARC DRAWING ROUTINE
4010 PRINT "~F~C";
4020 PRINT USING "#";NW(I,5);
4030 PRINT "<";
4040 PLOT NW(I,1)-5,NW(I,2)-5,NW(I,1)+5,NW(I,2)+5
4050 PRINT "~L+";
4060 PLOT NW(I,3)-5,NW(I,4)-5,NW(I,3)+5,NW(I,4)+5
4070 PRINT "'";
4080 PLOT NW(I,1),NW(I,2),NW(I,3),NW(I,4)
4090 RETURN
5000 REM *** PLOT REGION HIT ROUTINE
5010 IF F=3 OR F=4 OR F=6 GOTO 1070
5020 IF F=1 GOTO 7000
5030 FOR I=1 TO MN
5040 IF ABS(X-NW(I,1))<8 AND ABS(Y-NW(I,2))<8 THEN 5100
5050 IF ABS(X-NW(I,3))<8 AND ABS(Y-NW(I,4))<8 THEN 5100
5060 NEXT I
5070 GOTO 1070
5100 IF F=2 THEN GOTO 5110
5105 NW(I,5)=CR
5107 GOTO 5130
5110 NW(I,5)=0
5130 GOSUB 4000
5140 PRINT CHR$(7);
5150 GOTO 1070
7000 REM ADD ARC
7010 IF B1<>0 THEN GOTO 7100
7020 B1=X
```

```
7030 E1=Y
7040 PRINT CHR$(7);
7050 GOTO 1070
7100 MN=MN+1
7110 I=MN
7120 PRINT CHR$(7);
7150 NW(I,1)=S1:NW(I,2)=E1:NW(I,3)=X:NW(I,4)=Y:NW(I,5)=CR
7160 GOSUB 4000
7170 B1=0
7180 GOTO 1070
8000 REM *** SAVE ROUTINE
8010 DOS"OPEN 6 W *.DOS"
8012 MT=0
8014 FOR I=1 TO MN
8016 IF NW(I,5) <> 0 THEN MT=MT+1
8018 NEXT I
8020 PRINT #6: USING "##"; MT
8030 IF MN=0 THEN GOTO 8120
8040 FOR I=1 TO MN
8050 IF NW(I,5)=0 THEN GOTO 8110
8060 FOR J = 1 TO 4
8070 PRINT #6: USING "###"; NW(I,J):
8080 PRINT #6; ",":
8090 NEXT J
8100 PRINT #6: USING "#"; NW(I,5)
8110 NEXT I
8115 PRINT #6: CHR$(30)
8120 DOS "CLOSE 6 NETSAVE"
8125 PRINT CHR$(27):"OA1~W1,401349510~:8":CHR$(12):CHR$(27):"OA0~:F":
8130 RETURN
9000 REM *** REDRAW ROUTINE
9010 PRINT CHR$(27):"OA1~W1,1,510399":CHR$(12):CHR$(27):"OA0";
9020 DOS"OPEN 6 R NETSAVE"
9030 INPUT #6: MN
9040 IF MN=0 THEN GOTO 9090
9050 FOR I = 1 TO MN
9060 INPUT #6: NW(I,1),NW(I,2),NW(I,3),NW(I,4),NW(I,5)
9070 GOSUB 4000
9080 NEXT I
9090 DOS "CLOSE 6"
9095 PRINT CHR$(27):"OA1~W1,401349510~:8":CHR$(12):CHR$(27):"OA0~:F":
9100 RETURN
9500 REM *** CLEAR ROUTINE
9510 MN=0
9520 PRINT CHR$(27):"OA1~W1,1,510399":CHR$(12):CHR$(27):"OA0":
9525 PRINT CHR$(27):"OA1~W1,401349510~:8":CHR$(12):CHR$(27):"OA0~:F":
9530 RETURN
```

```
6,7,    NETWORK GENERATION PRIMARY MENU
5,430,45,480,7
55,430,95,480,7
105,430,145,480,7
155,430,195,480,7
205,430,245,480,7
255,430,295,480,7
16,425,2,"ADD"
57,425,4,"DELETE"
112,425,6,"SAVE"
150,425,3,"REDRAW"
200,425,5,"CHANGE"
262,425,1,"CLEAR"
70,502,7,"DIAL-A-RIDE NETWORK GENERATION"
#C4NETWORK GENERATION PRIMARY MENU C2
```

B-2

Chromatics (BASIC)
Display and Data Manipulation
Programs and Menu Data Sets for
Dial-A-Ride

```
50 CLEAR 400
52 RP=0
94 PRINT CHR$(27);"OA4";CHR$(27);"IA4";CHR$(27);"RO-";
95 OM=10
100 PRINT "~E";CHR$(12);"~C7,0,(,511511[illegible],4([illegible]1[illegible]0274040350511";
110 F$="MENU";GOSUB 4300
115 M=1:F1=0:F2=0:GOSUB [illegible]
120 M=1:F1=0:F2=0
130 GOSUB 4500
135 OUT&H90,0
140 ON ERROR#2 GOTO 500
150 OUT&H90,0
160 GOTO 150
500 REM *** MAIN MENU ROUTINE
505 X=CURSX(4):Y=CURSY(4)
510 GOSUB 4000
520 IF MI=0 THEN RESUME 150
530 F1=MI
540 F2=0
560 PRINT "~W1,401349510";CHR$(12);
562 M=0
564 IF F1<>4 THEN GOSUB 4500
570 M=F1+1
575 IF F1=4 THEN M=M+1
580 GOSUB 4500
590 IF F1=1 THEN GOTO 5900
592 IF F1=2 THEN GOTO 6900
593 IF F1=3 THEN GOTO 7900
594 IF F1=4 THEN GOTO 9000
595 OUT&H90,0
600 RESUME
1000 REM ***  MID-RUN RESTART
1005 CLEAR 400
1010 RP=1
1020 GOTO 94
2000 REM *** TRANSMISSION ROUTINES
2100 REM *** XMIT/INIT
2110 ON ERROR#0 GOTO 0
2199 RETURN
2200 REM *** XMIT/COMPLETE
2210 PRINT CHR$(27);"CAF";CHR$(27);"IA4";
2215 Z=0 'THIS IS A DUMMY STATEMENT
2220 PRINT CHR$(27);"OA0";CHR$(27);"IA0";
2230 RETURN
2300 REM *** RECEIVE/INIT
2310 ON ERROR#0 GOTO 0
2320 PRINT CHR$(27);"CAF";CHR$(27);"IA4";
2330 Z=0  ' THIS IS A DUMMY STATMENT
2340 PRINT CHR$(27);"IA0";
2350 RETURN
2400 REM *** RECEIVE/COMPLETE
2410 PRINT CHR$(27);"IA4";
2420 PRINT CHR$(27);"IA0";CHR$(27);"OA0";
2430 RETURN
2500 REM *** TALLY ACTIVE TRIPS
2510 TY=0
2520 FOR L=1 TO NT
2530 IF TA(L)=1 THEN TY=TY+1
2540 NEXT L
2550 RETURN
2600 REM *** XMIT ACTIVE TRIP NUMBERS
2610 FOR  L=1 TO NT
2620 IF TA(L)=0 THEN 2650
2630 PRINT #4;",";
```

```
2640 PRINT #4:USING ",##":L;
2650 NEXT L
2660 RETURN
2670 REM *** TALLY ACTIVE CLUSTER TRIPS
2675 TY=0
2680 FOR L=1 TO NT
2685 IF TA(L)=1 AND TC(L,2)=CN THEN TY=TY+1
2690 NEXT L
2695 RETURN
2700 REM *** XMIT ACTIVE CLUSTER TRIP NUMBERS
2705 FOR L=1 TO NT
2710 IF TA(L)=0 OR TC(L,2)<>CN THEN 2725
2715 PRINT #4:",";
2720 PRINT #4: USING "##";L;
2725 NEXT L
2727 PRINT CHR$(21);"~C6ACTIVE CLUSTER XMITTED"
2730 RETURN
2800 REM *** XMIT NETWORK ON INITIATE
2802 PRINT "~W351401510510";CHR$(12);CHR$(21);"~C4BEGIN XMIT ";NT;" TRIPS"
2805 GOSUB 2100
2810 PRINT #4:1
2812 PRINT "~?005";
2815 GOSUB 2200
2820 ON ERROR#3 GOTO 2830
2825 GOTO 2825
2830 RESUME 2835
2835 GOSUB 2100
2840 PRINT #4: 1;",";NT;",";
2845 FOR I=1 TO NT
2850 FOR J=1 TO 4
2852 PRINT #4: USING "###"; TP(I,J);
2854 XX=I MOD 5:IF J=4 AND (XX=0 OR I=NT) THEN 2856
2855 PRINT #4: ",";:GOTO 2857
2856 PRINT #4: " "
2857 NEXT J
2862 IF XX<>0 AND I<>NT THEN NEXT I
2863 PRINT "~?005";
2865 GOSUB 2200
2870 ON ERROR#3 GOTO 2880
2875 GOTO 2875
2880 RESUME 2885
2885 ON ERROR#0 GOTO 0
2887 PRINT "~C6TRIP ";I;" XMITTED"
2890 NEXT I
2895 PRINT "~C4XMIT COMPLETE~?010";CHR$(12);"~G";
2897 ON ERROR#0 GOTO 0
2900 RETURN
3000 REM *** INITIALIZE
3010 DOS"OPEN 5 R NETSAVE"
3020 INPUT #5: NT
3030 DIM TP(NT,4),TC(NT,2),TA(NT),CK(20)
3040 FOR I=1 TO NT
3050 INPUT #5:TP(I,1),TP(I,2),TP(I,3),TP(I,4),TC(I,1)
3060 TC(I,2)=TC(I,1)
3070 NEXT I
3080 DOS"CLOSE 5"
3085 IF RR=0 THEN GOSUB 2000
3087 PRINT "~G";
3090 PRINT "~W0,0,511511~:8";CHR$(12);"~:F";
3095 GOSUB 5100
3100 RETURN
3200 REM *** MACRO - LOCATE TRIP
3210 TN=0:MT=0
3220 FOR I=1 TO NT
```

```
3230 IF ABS(X-TP(I,1))<5 AND ABS(Y-TP(I,2))<5 THEN TN=I:GOTO 3250
3240 IF ABS(X-TP(I,3))>5 OR  ABS(Y-TP(I,4))>5 THEN 3260
3245 TN=-I
3250 TN=I:PRINT CHR$(7):RETURN
3260 NEXT I
3270 RETURN
3400 REM *** MACRO - DRAW TRIP (TRIP I, COLOR C)
3410 PRINT "~C";
3420 PRINT USING "#";C;
3430 PRINT "~F+";
3440 PLOT TP(I,1)-5,TP(I,2)-5,TP(I,1)+5,TP(I,2)+5
3450 PRINT "~L+";
3460 PLOT TP(I,3)-5,TP(I,4)-5,TP(I,3)+5,TP(I,4)+5
3470 PRINT "~";
3480 PLOT TP(I,1),TP(I,2),TP(I,3),TP(I,4)
3490 RETURN
3600 REM *** MACRO - DRAW CLUSTER (CLUSTER CN, COLOR C)
3605 AT=0
3610 PRINT "~C";
3620 PRINT USING "#";C;
3630 FOR I=1 TO NT
3640 IF TC(I,2)<>CN THEN 3670
3655 IF C=0 THEN TA(I)=-1
3660 IF TA(I)<>0 THEN GOSUB 3400:AT=AT+1
3665 IF TA(I)=-1 THEN TA(I)=0
3670 NEXT I
3680 RETURN
3800 REM *** MACRO - LOCATE CLUSTER
3810 CN=0
3820 IF Y>400 THEN GOTO 3900
3830 GOSUB 3200
3840 IF TN=0 THEN RETURN
3850 CN=TC(TN,2)
3855 C=TC(TN,1)
3860 RETURN
3900 REM LOCATE CLUSTER FROM COLOR
3910 K=K+1
3920 IF K>NT THEN K=0:MJ=0:RETURN
3930 IF TC(K,1)<>C OR TA(K)=0 THEN 3910
3935 IF MJ=0 THEN 3970
3940 FOR J=1 TO MJ
3950 IF TC(K,2)=CK(J) THEN GOTO 3910
3960 NEXT J
3970 MJ=MJ+1:CK(MJ)=TC(K,2)
3980 CN=TC(K,2)
3990 C=TC(K,1)
3995 RETURN
4000 REM *** MACRO - LOCATE ITEM ON MENU (MENU M)
4010 MI=0
4020 FOR I=1 TO MX(M)
4030 IF X<MENU(M,I,1) OR X>MENU(M,I,3) THEN 4060
4040 IF Y<MENU(M,I,2) OR Y>MENU(M,I,4) THEN 4060
4050 GOTO 4080
4060 NEXT I
4070 RETURN
4080 MI=I:PRINT CHR$(7)
4082 IF M<>0 THEN 4088
4084 PRINT "~:8~H350400511511";CHR$(12);"~:F";
4086 GOTO 4090
4088 PRINT "~:8~H1,400350511";CHR$(12);"~:F";
4090 PRINT "~L~C7+~1";
4100 PLOT MENU(M,I,1),MENU(M,I,2),MENU(M,I,3),MENU(M,I,4)
4110 PRINT "~2";
4120 RETURN
```

```
4300 REM *** COLOR MENU READ-IN ROUTINE
4302 PRINT CHR$(21)
4310 DOS"OPEN 5 R "+F$
4320 INPUT #5: MM
4330 DIM MENU(MM,[illegible],5),M$(MM,12),HOMENU(MM,12,3),MX(MM),SMX(MM)
4340 FOR K=0 TO MM
4350 INPUT #5: MX(K),SMX(K)
4360 XX=MX(K)
4370 FOR I=1 TO XX
4380 INPUT #5: MENU(K,I,1),MENU(K,I,2),MENU(K,I,3),MENU(K,I,4),MENU(K,I,5)
4390 NEXT I
4400 XX=SMX(K)
4410 IF XX = 0 THEN 4460
4420 FOR I = 1 TO XX
4430 INPUT #5: HOMENU(K,I,1),HOMENU(K,I,2),HOMENU(K,I,3),M$(K,I)
4440 NEXT I
4460 NEXT K
4470 DOS"CLOSE 5"
4480 RETURN
4500 REM *** MENU DRAW ROUTINE
4510 PRINT "~G~K";
4520 FOR I=1 TO MX(M)
4530 PRINT "~F~C";
4540 PRINT USING "#";MENU(M,I,5);
4550 PRINT "$";
4560 PLOT MENU(M,I,1),MENU(M,I,2),MENU(M,I,3),MENU(M,I,4)
4570 PRINT "~C7~L$";
4580 PLOT MENU(M,I,1),MENU(M,I,2),MENU(M,I,3),MENU(M,I,4)
4590 NEXT I
4600 PRINT CHR$(21);
4610 IF SMX(M)=0 THEN RETURN
4620 FOR I = 1 TO SMX(M)
4630 PRINT "~G~U";
4640 PLOT HOMENU(M,I,1),HOMENU(M,I,2)
4650 PRINT "~C";
4660 PRINT USING "#"; HOMENU(M,I,3);
4670 PRINT CHR$(21);M$(M,I);
4680 NEXT I
4685 PRINT "~G";
4690 RETURN
5000 REM **** EDIT FUNCTIONS
5020 IF Y<400 THEN 5060
5030 IF X<350 THEN K=0:MJ=0:PRINT "~;8~W350400511511";CHR$(12);"~;F";
5040 IF X<350 THEN M=2:GOSUB 4000:F2=MI
5050 IF X>350 THEN M=0:GOSUB 4000:C=MI-1
5060 ON F2 GOTO 5100,5400,5200,5600
5070 IF F2=0 THEN 5900
5080 PRINT "~W1,401510510";CHR$(12);"~C7;350400350511";
5085 PRINT "~W1,1,510400~;8";CHR$(12);"~;F";
5090 F2=0:F1=0:OUT%H20,0: RESUME 120
5100 REM *** EDIT - RESTORE
5110 PRINT "~W1,1,510399~;F";CHR$(12);
5120 FOR I=1 TO NT
5130 TA(I)=1
5140 C=TC(I,1)
5150 GOSUB 3400
5160 NEXT I
5170 PRINT "~W1,1,511511~;8";CHR$(12);"~;F";
5180 F2=0
5185 IF F1=0 THEN RETURN
5190 GOTO 5900
5200 REM *** EDIT - SUBCUE ARC
5210 IF Y<400 THEN 5230
5215 PRINT "~;8~W1,1,511400";CHR$(12);"~;F";
```

```
5220 GOTO 5900
5230 PRINT "~:2~W1,1,511400";CHR$(12);"~:F";
5240 GOSUB 3200
5250 IF TN=0 THEN 5900
5260 TA(TN)=0
5270 C=0:I=TN
5280 GOSUB 3400
5290 GOTO 5900
5400 REM *** EDIT - SUBSET CLUSTER
5410 IF Y<400 OR X>350 THEN 5500
5420 K=0
5430 PRINT "~:2~W1,1,511400";CHR$(12);"~:F";
5440 GOTO 5900
5500 GOSUB 3800
5510 IF CN=0 THEN 5900
5520 C=0
5530 GOSUB 3600
5540 GOTO 5900
5600 REM *** EDIT - HIGHLIGHT CLUSTER
5610 IF Y<400 OR X>350 THEN 5700
5615 K=0
5620 PRINT "~:8~W1,1,511400";CHR$(12);"~:F";
5625 MJ=0
5630 GOTO 5900
5700 GOSUB 3800
5710 IF CN=0 THEN 5900
5720 PRINT "~:8~W1,1,511400";CHR$(12);"~:F~1";
5730 GOSUB 3600
5740 PRINT "~2";
5750 GOTO 5900
5900 REM *** EDIT - RESET LIGHT PEN AND RESUME EDIT
5905 OUT&H90,0
5910 RESUME 5920
5920 ON ERROR52 GOTO 5950
5930 OUT&H90,0
5940 GOTO 5940
5950 X=CURSX(4):Y=CURSY(4)
5960 GOTO 5000
6000 REM ***** CLUSTER FUNCTIONS
6010 IF Y<400 THEN 6050
6020 IF X<350 THEN K=0:MJ=0:PRINT "~:8~W350400511511";CHR$(12);"~:F";
6030 IF X<350 THEN M=3:GOSUB 4000:F2=MI
6040 IF X>350 THEN M=0:GOSUB 4000:C=MI-1
6050 ON F2 GOTO 6200,6300,6400,6500,6600,6700
6060 IF F2=0 THEN 6900
6070 PRINT "~W1,401510510";CHR$(12);"~C7*350400350511";
6080 PRINT "~W1,1,510400~:8";CHR$(12);"~:F";
6090 F2=0:F1=0:OUT&H90,0:RESUME 120
6200 REM *** CLUSTER - SELECT CLUSTER
6210 IF Y < 400 OR X>350 THEN 6250
6220 K=0:MJ=0:CC=0:OA=0
6230 PRINT "~:8~W1,1,511400";CHR$(12);"~W350400511511";CHR$(12);"~:F";
6240 GOTO 6900
6250 GOSUB 3800
6260 IF CN=0 THEN 6900
6265 OA=CN:CC=C
6270 PRINT "~:8~W1,1,511400";CHR$(12);"~:F~1";
6280 GOSUB 3600
6290 PRINT "~2";
6295 GOTO 6900
6300 REM *** CLUSTER - CREATE CLUSTER
6310 IF Y<400 THEN 6900
6320 IF X>350 THEN 6350
6330 GOTO 6900
```

```
6350 PRINT "~:8~W1,1,511400";CHR$(12);"~:F":
6360 OC=0:CA=CM+1:OM=OM+1
6370 GOTO 6900
6400 REM *** CLUSTER - ADD TO CLUSTER
6415 IF OC=0 THEN PRINT "~:8~W1,400380511";CHR$(12);"~:F"::F2=0:GOTO 6900
6417 IF Y<400 THEN 6430
6418 PRINT "~:8~W350400511511";CHR$(12);"~:F":
6420 GOTO 6900
6430 GOSUB 3200
6440 IF TN=0 THEN 6900
6450 TC(TN,1)=CC:TC(TN,2)=OA
6460 I=TN:C=CC:CN=OA
6465 PRINT "~1":
6470 GOSUB 3400
6475 PRINT "~2":
6480 GOTO 6900
6500 REM *** CLUSTER - DELETE FROM CLUSTER
6510 IF OC=0 THEN PRINT "~:8~W1,400380511";CHR$(12);"~:F"::F2=0:GOTO 6900
6520 IF Y<400 THEN 6540
6525 PRINT "~:8~W350400511511";CHR$(12);"~:F":
6530 GOTO 6900
6540 GOSUB 3200
6550 IF TN=0 THEN 6900
6555 IF TC(TN,2)<>OA THEN 6900
6560 TC(TN,1)=7:TC(TN,2)=9999:I=TN:C=7
6570 GOSUB 3400
6580 GOTO 6900
6600 REM *** REQUEST CLUSTER INFO
6602 PRINT "~W1,1,510400~:8";CHR$(12);"~:F":
6605 PRINT "~W351401510510";CHR$(12);CHR$(21);"~C4BEGIN CLUSTER REQUEST"
6606 OUT&H90,0:RESUME 6610
6610 GOSUB 2100:PRINT #4:2
6611 PRINT "~?005"::GOSUB 2200
6612 ON ERROR#3 GOTO 6614
6613 GOTO 6613
6614 RESUME 6615
6615 GOSUB 2100
6617 INPUT "DESIRED NO OF CLSTRS";DC
6620 INPUT "VEHICLE CAPACITY";VC
6625 YC=0     'DUMMY CAPACITY = 0
6626 GOSUB 2500
6627 IF DC*VC < TY THEN PRINT"~C7INSUFF CAP. FOR":TY:"TRIPS":GOTO 6617
6630 PRINT#4:3:",":DC:",":TY:",":VC:",":YC::GOSUB 2600
6635 PRINT #4:" ":PRINT "TRIP INFORMATION XMITTED"
6640 DIM S#(TY),T#(TY)
6645 FOR J=1 TO TY
6649 GOSUB 2200:GOSUB 2200
6650 INPUT #4: S#(J),T#(J):I=T#(J):TC(I,1)=S#(J) MOD 6:TC(I,2)=S#(J)+OM
6652 IF TC(I,1)=0 THEN TC(I,1)=6
6653 PRINT "~C3CLUS#":TC(I,2):"CLR=":TC(I,1):"TRP=":I
6655 NEXT J
6660 PRINT "~C4CLUSTER INFO RETURNED~C~W1,1,510399"
6665 FOR J=1 TO TY
6670 I=T#(J):C=TC(I,1):GOSUB 3400
6675 NEXT J
6680 OM=OM+DC:GOSUB 2200:ERASE S#,T#
6685 PRINT "~W351401510510";CHR$(12):M=0:GOSUB 4500
6687 PRINT "~G";
6688 PRINT "~W1,1,510510~:8";CHR$(12);"~:7":
6689 CN=0
6690 OUT&H90,0:GOTO 6930
6700 REM *** TRANSMIT CLUSTER INFO
6705 PRINT "~W351401510510";CHR$(12);CHR$(21);"~C4BEGIN CLUSTER XMIT"
6707 IF CN=0 THEN 6731
```

```
6705 GOSUB 2020:IF TY=0 THEN 6731
6710 OUT&H90,0:GOTO 6715
6715 GOSUB 2100:P [illegible] #4;3
6720 PRINT "~?0.0";:GOSUB 2200
6725 ON ERROR63 GOTO 6735
6730 GOTO 6730
6731 PRINT "~CONO CLUSTER SELECTED~?030"
6732 OUT&H90,0:RESUME 6785
6735 RESUME 6740
6740 GOSUB 2100,
6750 PRINT #4:TY;",0,0,2";:GOSUB 2700:PRINT #4:" ":PRINT "~?005";
6755 GOSUB 2200
6760 ON ERROR63 GOTO 6770
6765 GOTO 6765
6770 RESUME 6775
6775 GOSUB 2100:GOSUB 2200
6780 PRINT "~OFKNIT ACKNOWLEDGED~?020";GOSUB 2200
6785 PRINT "~W351401510510";CHR$(12);K=0:GOSUB 4500:PRINT "~G";
6786 PRINT "~W1,1,510510~:8";CHR$(12);"~:F";
6787 CN=0
6790 OUT&H90,0:GOTO 6530
6900 REM *** CLUSTER - RESET LIGHT PEN AND RESUME CLUSTER
6910 OUT&H90,0
6920 RESUME 6930
6930 ON ERROR62 GOTO 6960
6940 OUT&H90,0
6950 GOTO 6950
6960 X=CURSX(4):Y=CURSY(4)
6970 GOTO 6000
7000 REM ***** ROUTE FUNCTIONS
7010 IF Y<400 THEN 7200
7020 IF X<350 THEN K=0:MJ=0:PRINT "~:8~W350400511511";CHR$(12);"~F":
7030 IF X<350 THEN M=4:GOSUB 4000:F2=M1
7035 IF X>350 THEN M=0:GOSUB 4000:C=MJ-1
7040 ON F2 GOTO 7200,3000
7050 IF F2=0 THEN 7900
7060 PRINT "~W1,401510510";CHR$(12);"~07~G";350400351511";
7070 PRINT "~W1,1,510399~:8";CHR$(12);"~:F";
7080 F2=0:F1=0:OUT&H90,0: RESUME 120
7200 REM *** ROUTE - SELECT CLUSTER
7210 IF Y<400 OR X>350 THEN 7250
7220 K=0:MJ=0
7230 PRINT "~:8~W1,1,510399";CHR$(12);"~:F";
7240 GOTO 7900
7250 GOSUB 3600
7260 IF CN=0 THEN 7900
7270 PRINT "~:8~W1,1,510399";CHR$(12);"~:F~1";
7280 GOSUB 3600
7290 PRINT "~2";
7295 GOTO 7900
7300 REM *** ROUTE/SELECT REDRAW ACTIVE CLUSTERS
7305 PRINT "~W1,1,510399~:F";CHR$(12);
7310 FOR I=1 TO NT
7320 IF TA(I)=0 THEN 7350
7330 C=TC(I,1)
7340 GOSUB 3400
7350 NEXT  I
7360 PRINT "~W1,1,511511~:8";CHR$(12);"~:F";
7370 LR=0
7380 RETURN
7400 REM *** REQUEST ROUTE INFO
7405 PRINT "~W351401510510";CHR$(12);CHR$(28);"~CASLOIN ROUTE REQUEST".
7410 GOSUB 2270:IF CN=0 OR TY=0 THEN PRINT "~CANO CLUSTER SELECTED~?330":GOTO 7
7412 PRINT "~W1,1,510399~:0";CHR$(12);"~:F";"~W351,01510497";
```

```
7419 OUT&H90,0:RESUME 7420
7420 GOSUB 2100:PRINT#4;3:PRINT "~?005";:GOSUB 2200
7425 ON ERROR#3 GOTO 7435
7430 GOTO 7430
7435 RESUME 7440
7440 GOSUB 2100:GOSUB 2200
7445 PRINT #4;TY;",0,0,1";:GOSUB 2700:PRINT#4;" "
7455 FOR I = 1 TO 2*TY
7457 PRINT"~?001";:GOSUB 2200:GOSUB 2200
7460 INPUT#4;RT(I)
7465 IF RT(I)>0 THEN PRINT "~C3   PICKUP     ";RT(I)
7466 IF RT(I)<0 THEN PRINT "~C3   DELIVER    ";-RT(I)
7470 NEXT I
7475 TD=0:PRINT"~%0~C5~G";
7480 FOR I=1 TO 2*TY
7485 Z=RT(I)
7490 IF Z<0 THEN X2=TP(-Z,3):Y2=TP(-Z,4)
7495 IF Z>0 THEN X2=TP(Z,1):Y2=TP(Z,2)
7500 IF I=1 THEN 7515
7505 TD=TD+SQR((X1-X2)^2+(Y1-Y2)^2)
7510 PRINT "*";:PLOT X1,Y1,X2,Y2
7515 PRINT "~F*";:PLOT X2,Y2:PRINT "005~L"
7517 X1=X2:Y1=Y2
7520 NEXT I
7525 PRINT "~F~1*";:PLOT X2,Y2:PRINT "005~2~L~F";
7530 LA=2*TY:OUT&H90,0:GOTO 7532
7531 OUT&H90,0:RESUME 7532
7532 PRINT "~W351401510510~*F";CHR$(12);:N=5:GOSUB 4500
7535 GOSUB 7800:GOSUB 2200
7540 GOTO 8120
7600 REM *** ROUTE/ROUTE XMIT INFC
7605 PRINT "~W351401510510";CHR$(12);CHR$(21);"~C4BEGIN ROUTE XMIT"
7610 GOSUB 2670:IF CN=0 OR TY=0 THEN PRINT"~C6NO CLUSTER SELECTED~?030":GOTO 753
7615 IF LA<2*TY THEN PRINT "~C6INCOMPLETE ROUTE~?030":GOTO 7531
7620 OUT&H90,0:RESUME 7625
7625 GOSUB 2100:PRINT#4;6:PRINT "~?005";:GOSUB 2200
7630 ON ERROR#3 GOTO 7640
7635 GOTO 7635
7640 RESUME 7645
7645 GOSUB 2200:GOSUB 2100:PRINT #4;"1,";:PRINT #4; USING "#####";TD;
7647 PRINT#4;",";TY;:GOSUB 2700:PRINT#4;" ":PRINT "~?005";
7650 ON ERROR#3 GOTO 7660
7655 GOTO 7655
7660 RESUME 7662
7662 GOSUB 2200
7665 GOSUB 2200:PRINT "~C6XMIT ACKNOWLEDGED~?030"
7666 GOSUB 2200
7670 GOTO 7532
7800 REM *** ROUTE/ROUTE POST PARTIAL TOUR DISTANCE
7805 PRINT "~W350485510510~*F";CHR$(12);
7810 PRINT "~C7~U360497";CHR$(21);"TOTAL DISTANCE =";:PRINT USING "#####";TD;
7820 PRINT "~G";
7830 RETURN
7900 REM *** ROUTE - RESET LIGHT PEN AND RESUME EDIT
7910 OUT&H90,0
7920 RESUME 7930
7930 ON ERROR#2 GOTO 7960
7940 OUT&H90,0
7950 GOTO 7950
7960 X=CURSX(4):Y=CURSY(4)
7970 GOTO 7000
8000 REM *** ROUTE - ROUTE PACFC
8010 IF CN=0 THEN 7900
8020 PRINT "~W1,1,510399";CHR$(12);
```

```
8030 C=2:GOSUB 3600
8040 PRINT "~?351[illegible]1C10510";CHR$(12);:M=5:GOSUB 4500
8050 [illegible]=2*AT
8060 DIM RT(2*AT)
8070 LA=0:RB=0:[illegible]
8080 LR=1
8100 REM ROUTE/ROUTE RESET LIGHT PEN
8110 CURS[illegible]0,0
8115 [illegible] (120)
8120 ON LRPON%4 GOTO 8145
8130 CURSOR0,0
8140 GOTO 8140
8145 X=CURSX(4):Y=CURSY(4)
8150 IF Y<400 THEN 8500
8160 IF X<350 THEN 8100
8170 M=5:GOSUB 4000:F3=MI
8180 ON F3 GOTO 8200,8300,8700,7400,7600
8190 GOTO 8100
8200 REM ROUTE/ROUTE START
8205 IF RB=-1 THEN PRINT "~!D~W1,1,510399";CHR$(12);
8210 RB=-1:TD=0
8220 LA=0:PRINT "~!C~W1,1,510,399";CHR$(12);
8230 GOTO 8100
8300 REM ROUTE/ROUTE BACK-UP
8310 PRINT "~:8~W3F0400511511";CHR$(12);"~!F";
8320 IF LA<2 THEN 8100
8330 A1=ABS(RT(LA)):H1=A1/RT(LA)
8340 A2=ABS(RT(LA-1)):H2=A2/RT(LA-1)
8350 IF H1>0 THEN X1=TP(A1,1):Y1=TP(A1,2)
8360 IF H1<0 THEN X1=TP(A1,3):Y1=TP(A1,4)
8370 IF H2>0 THEN X2=TP(A2,1):Y2=TP(A2,2)
8380 IF H2<0 THEN X2=TP(A2,3):Y2=TP(A2,4)
8390 PRINT "~CC~!C";:PLOT X1,Y1,Y2,Y2:PRINT "~F";:PLOT X1,Y1:PRINT "005~L~!F"
8392 TD=TD-SQR((X1-X2)^2+(Y1-Y2)^2)
8393 GOSUB 7200
8395 LA=LA-1:GOTO 8655
8400 REM *** ROUTE/ROUTE STARTING POINT GIVEN
8410 LA=0
8430 GOSUB 3200
8440 IF TN=0 THEN 8100
8450 IF TC(TN,2)<>CN OR TA(TN)=0 THEN 8100
8460 IF HT<0 THEN 8100
8470 LA=1:RT(1)=TN:RB=0
8475 PRINT "~:8~W3500400510510";CHR$(12);
8480 PRINT "~!C~W1,1,510399";CHR$(12);
8490 TD=0
8495 X2=TP(TN,1):Y2=TP(TN,2):GOTO 8655
8500 REM *** ROUTE/ROUTE NEXT POINT
8510 IF RB=-1 THEN 8400
8520 IF LA=0 THEN 8100
8540 GOSUB 3200
8550 IF TN=0 THEN 8100
8560 IF TC(TN,2) <> CN OR TA(TN)=0 THEN 8100
8562 PRINT "~:8~W1,1,510399";CHR$(12);"~!F";
8565 FG=0
8570 FOR J=1 TO LA
8580 IF RT(J)=TN*HT THEN 8100
8590 IF RT(J)=-TN*HT THEN FG=1
8600 NEXT J
8605 IF FG=0 AND HT<0 THEN 8100
8610 LA=LA+1
8620 RT(LA)=TN*HT
8625 A1=ABS(RT(LA-1)):H1=A1/RT(LA-1)
8630 IF H1>0 THEN X1=TP(A1,1):Y1=TP(A1,2)
```

```
8630 IF H1<0 THEN X1=TP(A1,3):Y1=TP(A1,4)
8640 IF HT>0 THEN X2=TP(TN,3):Y2=TP(TN,2)
8645 IF HT<0 THEN X2=TP(TN,3):Y2=TP(TN,4)
8647 TD=TD+SQR((X1-X2)^2+(Y1-Y2)^2)
8650 PRINT "~:C~CA^";:PLOT X1,Y1,X2,Y2
8655 PRINT "~:C~C^1~F^";:PLOT X2,Y2:PRINT "@CF~J~L~:F";
8656 GOSUB 7800
8660 GOTO 6100
8700 REM ROUTE/ROUTE STOP
8705 PRINT "~:A~U050400S1512":CHR$(12);"~:F";
8720 PRINT "~W3514015105:0":CHR$(12);:N=0:GOSUB 4500
8722 PRINT "~G";
8725 ERASE RT
8727 GOSUB 7700
8730 GOTO 7900
9000 REM ***** COVER SUBFUNCTION
9010 OUT&H90,0
9020 RESUME 9040
9030 PRINT CHR$(27);"OA1";
9040 PRINT "~W1,1,510399~:F";CHR$(12);"~W3514015105:0";CHR$(12);
9050 PRINT CHR$(21);"~C4BEGIN COVER REQUEST"
9060 GOSUB 2100:PRINT#4;7
9070 PRINT "~?005";:GOSUB 2200
9080 ON ERROR#3 GOTO 9100
9090 GOTO 9390
9100 RESUME 9110
9110 GOSUB 2100
9115 PRINT #4;1:PRINT "~?005";:GOSUB 2200
9120 PRINT CHR$(27);"OA1";"~W3514015105:0~C6COVER REQUEST XMITTED~?005"
9130 GOSUB 2200
9135 PRINT CHR$(27);"OA1~C5";
9140 INPUT#4;NC
9150 PRINT NC;" CLUSTERS RETURNED"
9160 FOR K = 1 TO NC
9164 C=K MOD 6:IF C=0 THEN C=6
9170 GOSUB 2200
9180 INPUT#4;RT
9185 PRINT CHR$(27);"OA1";CHR$(21);
9190 PRINT RT;" TRIPS IN CLUSTER";K
9195 PRINT CHR$(27);"OA0";
9200 FOR J=1 TO RT
9210 GOSUB 2200
9220 INPUT#4;I
9230 TC(I,1)=C:TC(I,2)=K:PRINT "~G";:GOSUB 3400:TA(I)=1
9240 NEXT J
9245 PRINT CHR$(21);CHR$(27);"OA1";
9250 NEXT K
9260 PRINT CHR$(27);"OA1~C4COVER REQUEST COMPLETE~?030"
9280 OM=NC+1
9290 PRINT CHR$(12);CHR$(27);"OA0~G";
9300 REM *** COVER MENU SERVICE ROUTINE
9310 OUT&H90,5
9320 ON ERROR#2 GOTO 9340
9330 GOTO 9330
9340 X=CURSX(4):Y=CURSY(4)
9350 GOSUB 4000
9355 OUT&H90,0
9360 IF MI=0 THEN RESUME 9300
9365 IF MI=1 THEN RESUME 10000
9370 IF MI=2 THEN 9000
9380 RESUME 9300
9400 REM *** COVER/EXIT
9410 PRINT "~:1,401510510":CHR$(12);"~CY"3906803?0511";
9420 F1=0:F2=0: RESUME 120
```

```
10000 REM *** CHAINING INITIALIZATION
10010 PRINT CHR$(21);"~P~W0,0,511511";CHR$(12);"~U~X4,~Y4,ENO CLUSTERING"
10020 PRINT "BEGIN CHAINING"
10025 ERASE MTIG,NU,MENU,MX,SMY
10030 OM=0:LC=0
10040 F$="MENU7":GOSUB 4300
10050 PRINT "CHAINING MENU READ IN~Y30~";CHR$(12);"~X1,~Y1,~";
10060 PRINT "~A40,0,511511~0,4005;140;7540;~8511~";
10070 F1=0:M=2:GOSUB 4500
10072 PRINT CHR$(21);CHR$(27);"OA1~W351401511510";
10080 GOSUB 10900
10085 PRINT CHR$(27);"OA0~G~W351401510510";CHR$(12);:M=0:GOSUB 4500
10087 IF F1<>1 THEN GOSUB 10300:F1=0
10089 PRINT "~G";
10090 OUT&H90,0
10100 ON ERROR 2 GOTO 10140
10110 OUT&H90,0
10120 GOTO 10120
10140 X=CURSX(4):Y=CURSY(4):GOTO 11000
10300 REM *** CHAINING - RESTORE CLUSTERED ARCS
10305 PRINT "~A1,1,510399";CHR$(12);
10310 FOR K=1 TO NC
10320 CT(K,5)=7:CT(K,6)=0:C=7
10330 IF F2<>6 OR F2=6 AND TA(K)=1 THEN TA(K)=1:GOSUB 10800
10340 NEXT K
10350 OM=LC
10360 IF LC=0 THEN RETURN
10370 FOR K=1 TO LC
10380 FOR J=1 TO 6
10390 CH(K,J)=CH(CL(K),J)
10400 NEXT J
10410 CL(K)=K
10420 NEXT K
10430 RETURN
10500 REM *** CHAINING - SET UP CHAIN LISTS
10505 OUT&H90,0
10510 GOSUB 2100
10520 PRINT#4;4:PRINT "~0005";:GOSUB 2200
10530 PRINT CHR$(27);"OA1~C6CHAINING REQUEST XMITTED~C3"
10540 INPUT#4;NC
10550 PRINT NC;" CLUSTERS RETURNED"
10560 DIM CT(NC,6),CH(25,6),CC(25),CL(25)
10570 FOR K=1 TO NC
10580 C=K MOD 6:IF C=0 THEN C=6
10600 GOSUB 2200
10610 INPUT#4;CT(K,1),CT(K,2),CT(K,3),CT(K,4)
10630 PRINT "~1";
10640 CT(K,5)=C
10650 CT(K,6)=0
10655 TA(K)=1
10660 GOSUB 10800
10670 PRINT "~2";
10680 GOSUB 2200
10690 INPUT#4;RT
10700 PRINT CHR$(27);"OA1";RT;" TRIPS IN CLUSTER";K
10710 FOR J=1 TO RT
10720 GOSUB 2200
10730 INPUT#4;I
10735 PRINT CHR$(27);"OA0";
10740 TC(I,1)=0:TC(I,2)=K:PRINT"~G";:GOSUB 3400:PRINT CHR$(27);"OA1";
10750 NEXT J
10760 NEXT K
10770 PRINT CHR$(27);"OA1~C4CHAIN INFO READ IN~C100"
10780 GOSUB 2200
```

```
10790 RETURN
10810 REM *** MACRO - DRAW CLUSTERED ARC (ARC K, COLOR C)
10815 PRINT "~C~C";
10820 PRINT USING "~";C;
10830 PRINT "~F";
10840 PLOT CT(K,1)-7,CT(K,2)-7,CT(K,1)+7,CT(K,2)+7
10850 PRINT "~L+";
10860 PLOT CT(K,3)-7,CT(K,4)-7,CT(K,3)+7,CT(K,4)+7
10870 PRINT "~";
10880 PLOT CT(K,1),CT(K,2),CT(K,3),CT(K,4)
10882 TL=SQR((CT(K,1)-CT(K,3))^2+(CT(K,2)-CT(K,4))^2)
10884 IF C=7 THEN CT(K,6)=0
10886 IF C=0 THEN TA(K)=0
10900 RETURN
11000 REM **** CHAINING FUNCTIONS
11010 IF Y<400 THEN 11050
11020 IF X<350 THEN L=0:MJ=0:PRINT "~;F~W350400511511";CHR$(12);"~;F";
11030 IF X<350 THEN M=2:GOSUB 4000:F2=MI
11040 IF X>350 THEN M=3:GOSUB 4000:C=MI-1
11050 ON F2 GOTO 11200,11400,11600,12000,12500
11060 OUT&H90,0
11070 RESUME 10090
11200 REM *** CHAINING - SELECT CHAIN
11210 IF Y<400 OR X>350 THEN 11250
11220 L=0:MJ=0:OO=0:OA=0
11230 PRINT "~;8~H1,1,511399";CHR$(12);"~W351400510510";CHR$(12);"~;F";
11240 OUT&H90,0:RESUME 10090
11250 GOSUB 13000
11260 IF CN=0 THEN OUT&H90,0:RESUME 10090
11270 OA=CN:CO=0
11280 PRINT "~;8~H1,1,511400";CHR$(12);"~;F~1";
11290 I=CN:GOSUB 13600
11300 PRINT "~2";
11310 OUT&H90,0:RESUME 10090
11400 REM *** CHAINING - EDIT SELECTION
11410 PRINT "~H1,401349510";CHR$(12);:M=3:GOSUB 4500
11415 F2=0
11420 OUT&H90,0
11430 RESUME 14000
11600 REM *** CHAINING - REQUEST CHAINS
11605 PRINT "~H1,1,510...";CHR$(12);"~;F";
11610 PRINT "~W351401510510";CHR$(12);CHR$(21);"~C4~GIN CHAIN REQUEST"
11615 OUT&H90,0:RESUME 11620
11620 GOSUB 2100:PRINT#4;5:PRINT"~?005";:GOSUB 2200
11625 ON ERROR63 GOTO 11635
11630 GOTO 11630
11635 RESUME 11640
11640 GOSUB 2100
11645 INPUT "MAXIMUM NO OF CHAINS";DC
11650 PRINT #4;DC;
11652 TY=0
11653 FOR I=1 TO NC
11654 IF TA(I)=1 THEN TY=TY+1
11655 NEXT I
11656 PRINT#4;",";TY;
11657 FOR I=1 TO NC
11660 IF TA(I)=1 THEN PRINT#4;",";:PRINT#4;USING "##";I;
11665 NEXT I
11667 PRINT#4;" "
11670 PRINT CHR$(27);"0A1";CHR$(21);"~W351401510510~C8ACTIVE CLUSTERS XMITTED"?
11675 GOSUB 2200:GOSUB 10500
11680 PRINT CHR$(27);"0A1~03";
11685 INPUT#4;NM
11690 PRINT NM;"CHAINS RETURNED"
```

```
11695 FOR K=1 TO AN
11700 C=K MOD 5:IF C=0 THEN C=5
11705 GOSUB 2200
11710 INPUT#4:RT
11715 PRINT CHR$(27):"OA1";CHR$(21);
11720 PRINT RT;" CLUSTERS IN CHAIN ";K
11725 PRINT CHR$(27);"OA0";
11730 CM=CM+1
11735 FOR J=1 TO 6
11740 CH(CM,J)=0
11745 NEXT J
11750 FOR J=1 TO RT
11755 GOSUB 2200
11760 INPUT#4:G
11765 CT(G,5)=C:CT(G,6)=CM:TA(G)=1
11770 IF J<7 THEN CH(CM,J)=G
11772 PRINT CHR$(27);"OA1  CHAIN TO ";G
11780 NEXT J
11785 LC=LC+1:CL(LC)=CM
11790 PRINT CHR$(27);"OA0~G";
11795 I=CM:GOSUB 13600
11800 PRINT CHR$(21);CHR$(27);"CA1";
11805 NEXT K
11810 PRINT CHR$(27);"OA1~C4CHAIN REQUEST COMPLETE~?030"
11815 PRINT CHR$(12);CHR$(27);"OA0~G";
11817 PRINT "~:8~W1,1,510510";CHR$(12);"~:F";
11820 F1=-1:GOTO 10085
12000 REM *** CHAIN - XMIT CHAIN
12005 PRINT "~W0514015105I0";CHR$(12);CHR$(21);"~C4BEGIN CHAIN XMIT"
12010 IF CN=0 THEN PRINT "NO CHAIN SELECTED~?020":OUT&H90,0:RESUME 12155
12015 TY=0
12020 FOR J=1 TO 6
12025 IF CH(CN,J)=0 THEN 12052
12030 H=CH(CN,J)
12035 FOR G=1 TO NT
12040 IF TC(G,2)=H THEN TY=TY+1
12045 NEXT G
12050 NEXT J
12052 LC=LC+1:CL(LC)=CN
12055 PRINT TY;" TRIPS IN CHAIN"
12060 OUT&H90,0:RESUME 12065
12065 GOSUB 2100:PRINT#4:5:PRINT"~?005";:GOSUB 2200
12070 ON ERROR#3 GOTO 12080
12075 GOTO 12075
12080 RESUME 12082
12082 GOSUB 2200
12085 GOSUB 2100:PRINT#4:"1,";:PRINT#4:USING"#####";CC(CN);
12090 PRINT#4:",";TY;
12095 FOR J=1 TO 6
12100 IF CH(CN,J)=0 THEN 12130
12105 H=CH(CN,J)
12110 FOR G=1 TO NT
12115 IF TC(G,2)=H THEN PRINT#4:",";G;
12120 NEXT G
12125 NEXT J
12130 PRINT#4:" ":PRINT"~?005";:GOSUB 2200
12135 ON ERROR#3 GOTO 12145
12140 GOTO 12140
12145 RESUME 12147
12147 GOSUB 2200
12150 GOSUB 2200:PRINT "~C6XMIT ACKNOWLEDGED~?030"
12155 PRINT"~W0514015105I0~:F";CHR$(12);:M=0:GOSUB 4500
12157 PRINT "~G~:6~W1,1,510510";CHR$(12);"~:F";
12160 OUT&H90,0:GOTO 10090
```

```
12500 REM ***** FINALE *****
12510 PRINT "~U1,1 510510~F";CHR$(12);CHR$(27);"OA1~W351431518510";CHR$(12);
12520 PRINT CHR$(27);"OA1~CORIGIN FINALE"
12525 OUT&H80,0:RESUME 12527
12527 GOSUB 2100
12528 GOSUB 2200
12530 PRINT#4,7:PRINT"~7005"::GOSUB 2200
12535 PRINT CHR$(27);"OA1~CNCOLOR REQUEST XMITTED"
12540 ON ERROR GOTO 12560
12550 GOTO 12550
12560 RESUME 12565
12565 GOSUB 2100
12570 PRINT#4,12:PRINT"~7005"::GOSUB 2200
12580 PRINT CHR$(27);"OA1~CNFINALE REQUEST XMITTED"
12600 PRINT CHR$(27);"OA1~C3";
12610 INPUT#4:NH
12620 IF NH<0 THEN 12800
12630 PRINT CHR$(27);"OA1";NH;" CHAINS RETURNED"
12640 FOR H=1 TO NH
12650 C=H MOD 6:IF C=0 THEN C=6
12660 GOSUB 2200
12670 INPUT#4:RT
12675 PRINT CHR$(27);"OA1";CHR$(21);"CHAIN";H;"COLUMN";RT
12676 PRINT "    CHAIN LIST ELE";CL(RT)
12680 I=CL(RT):PRINT CHR$(27);"C/C~G";:GOSUB 13600
12682 Z=CL(RT)
12684 FOR J=1 TO 6
12686 IF CH(Z,J)<>0 THEN CT(CH(Z,J),5)=C:CT(CH(Z,J),6)=Z
12688 NEXT J
12690 NEXT H
12700 GOSUB 2200
12710 PRINT CHR$(27);"OA1~CAFINALE COMPLETE~?030"
12715 PRINT "~;U1,1,510510";CHR$(12);"~;F";
12720 F1=-1:GOTO 13005
12800 REM - UNCOVERED CLUSTERS
12810 PRINT CHR$(27);"OA1~C6UNCOVERED CLUSTER AREAS"
12820 NH=-NH
12830 FOR H=1 TO NH
12840 C=7
12850 GOSUB 2200
12860 INPUT#4:I
12870 PRINT"~G"::GOSUB 3400
12880 NEXT H
12890 GOSUB 2200
12895 PRINT "~7030";:GOSUB 10300
12900 GOTO 10025
13000 REM *** CHAINING MACRO - LOCATE CHAIN
13010 CN=0
13020 IF Y>400 THEN 13200
13030 GOSUB 13400
13040 IF TN=0 THEN RETURN
13050 CN=CT(TN,6)
13060 C=CT(TN,5)
13070 RETURN
13200 REM *** CHAINING MACRO - LOCATE CHAIN FROM COLOR
13210 L=L+1
13220 IF L>NC THEN L=0:MJ=0:RETURN
13230 IF CT(L,5)<>C OR CT(L,6)=0 OR TA(L)=0 THEN 13210
13240 IF MJ=0 THEN 13280
13250 FOR J=1 TO MJ
13260 IF CT(L,6)=CK(J) THEN 13210
13270 NEXT J
13280 MJ=MJ+1:CK(MJ)=CT(L,6)
13290 CN=CT(L,6)
```

```
13300 C=CT(I,5)
13310 RETURN
13400 REM *** CHAINING MACRO - LOCATE CLUSTER ITEM
13410 TN=0:HT=0
13420 FOR I=1 TO NC
13430 IF ABS(X-CT(I,1))<10 AND ABS(Y-CT(I,2))<10 THEN HT=1:GOTO 13460
13440 IF ABS(X-CT(I,3))>9 OR ABS(Y-CT(I,4))>9 THEN 13470
13450 HT=-1
13460 TN=I:CN=CT(I,6):C=CT(I,5):PRINT CHR$(7);:RETURN
13470 NEXT I
13480 RETURN
13600 REM *** MACRO - DRAW CHAIN I, COLOR C
13605 CC(I)=0:TL=0
13610 IF I=0 THEN RETURN
13620 L=1:PRINT"~C";:PRINT USING "#";C;
13630 K=CH(I,L):GOSUB 10800
13640 OX=CT(K,3):OY=CT(K,4)
13650 L=L+1
13660 IF L<7 THEN K=CH(I,L)
13670 IF L>6 OR K=0 THEN RETURN
13675 GOSUB 10800:CC(I)=CC(I)+TL
13680 NX=CT(K,1):NY=CT(K,2)
13682 T1=CT(K,3):T2=CT(K,4)
13685 IF C=7 THEN PRINT "~C";:PRINT USING "#";0;
13690 GOSUB 13800
13695 IF C=7 THEN PRINT "~C";:PRINT USING "#";7;
13697 OX=T1:OY=T2
13700 GOTO 13650
13800 REM *** DRAW DASHED LINE FROM (OX,OY) TO (NX,NY)
13810 TL=SQR((OX-NX)^2+(OY-NY)^2)
13815 CC(I)=CC(I)+TL
13820 DN=TL/6
13830 XD=NX-OX:XQ=XD/DN
13840 YD=NY-OY:YQ=YD/DN
13850 FOR K=1 TO DN/2
13860 PRINT "*";
13870 PLOT OX,OY,OX+XQ,OY+YQ
13880 OX=OX+2*XQ
13890 OY=OY+2*YQ
13900 NEXT K
13910 RETURN
14000 REM *** CHAINING - EDIT SUBFUNCTION
14010 OUT&H90,0
14020 ON ERROR#2 GOTO 14040
14030 GOTO 14030
14040 X=CURSX(4):Y=CURSY(4)
14050 IF Y<400 THEN 14090
14060 IF X<350 THEN L=0:MJ=0:PRINT "~:F~W350400511511";CHR$(12);"~:F"
14070 IF X<350 THEN M=3:GOSUB 4000:F2=MI
14080 IF X>350 THEN M=0:GOSUB 4000:C=MI-1
14090 ON F2 GOTO 14200,14400,14600,14800,15000,15200
14100 IF F2=0 THEN OUT&H90,0:RESUME 14000
14110 PRINT "~:S~W1,1,511511";CHR$(12);"~:F~W1,401349510";CHR$(12);
14120 F1=-1:M=2:GOSUB 4500:PRINT "~G";
14130 OUT&H90,0:RESUME 10065
14200 REM *** CHAIN/EDIT - SELECT CHAIN
14210 IF Y<400 OR X>350 THEN 14250
14220 L=0:MJ=0:CC=0:OA=0
14230 PRINT "~:S~W1,1,515399";CHR$(12);"~W351405510510";CHR$(12);"~:F";
14240 OUT&H90,0:RESUME 14000
14250 GOSUB 13000
14260 IF CN=0 THEN OUT&H90,0:RESUME 14000
14270 OA=CN:CC=C
14280 PRINT "~:S~W1,1,511400";CHR$(12);"~:F~1";
```

```
14250 I=CN:GOSUB 13600
14300 PRINT "~2";
14350 OUT&H90,0:RESUME 14000
14400 REM *** CHAIN/EDIT - DISCO CHAIN
14410 IF Y<400 OR X>350 THEN 14450
14420 L=0:MJ=0:OC=0:OA=0
14430 PRINT "~:~W1,1,510400":CHR$(12);"~;F";
14440 OUT&H90,0:RESUME 14000
14450 GOSUB 13000
14460 IF CN=0 THEN OUT&H90,0:RESUME 14000
14470 C=7:I=CN:GOSUB 13600
14480 OUT&H90,0:RESUME 14000
14600 REM *** CHAIN/EDIT - CREATE CHAIN
14605 IF Y>400 AND X>350 THEN OC=C
14610 IF Y<400 OR X>350 THEN OUT&H90,0:RESUME 14000
14620 PRINT "~;3~W1,1,510400":CHR$(12);"~;F";
14630 OC=C:OA=CM+1:OC(OA)=0
14640 FOR J=1 TO 6
14650 CH(CA,J)=0
14660 NEXT J
14670 OUT&H90,0:RESUME 14000
14800 REM *** CHAIN/EDIT - IGNORE CHAIN
14810 IF Y<400 OR X>350 THEN 14850
14820 L=0:MJ=0:OC=0:OA=0
14830 PRINT "~;8~W1,1,510399":CHR$(12);"~W351400510510":CHR$(12);"~;F";
14840 OUT&H90,0:RESUME 14000
14850 GOSUB 13000
14860 IF CN=0 THEN OUT&H90,0:RESUME 14000
14870 I=CN:C=0
14880 PRINT "~;8~W1,1,511400":CHR$(12);"~;F";
14890 GOSUB 13600
14900 OUT&H90,0
14910 RESUME 14000
15000 REM *** CHAIN/EDIT - ADD TO CHAIN
15010 IF OC=0 OR OA=0 THEN PRINT"~;8~W1,400350510~;F";:F2=0:OUT&H90,0:RESUME 140
15020 IF Y<400 THEN 15050
15030 PRINT "~;8~W350400511511":CHR$(12);"~;F";
15032 OM=CM+1
15033 FOR J=1 TO 6
15034 CH(CM,J)=CH(OA,J):IF CH(CA,J)<>0 THEN CT(CH(OA,J),6)=OM
15035 NEXT J
15036 OA=CM
15040 OUT&H90,0:RESUME 14000
15050 GOSUB 13400
15060 IF TN=0 OR CT(TN,6)<>0 THEN OUT&H90,0:RESUME 14000
15070 CT(TN,5)=OC:CT(TN,6)=OA
15080 FOR J=1 TO 6
15090 IF CH(CA,J)=0 THEN 15120
15100 NEXT J
15110 GOTO 15130
15120 CH(CA,J)=TN
15130 I=OA:C=OC
15140 PRINT "~1";:GOSUB 13600:PRINT "~2";
15150 OUT&H90,0:RESUME 14000
15200 REM *** CHAIN/EDIT - RESTORE
15205 GOSUB 10300
15207 PRINT "~;8~W1,1,510510":CHR$(12);"~;F";
15210 OUT&H90,0:RESUME 14000
31000 STOP
```

```
#,      C3DIAL-A-RIDE PRIMARY MENU FILE C2
#,8,     C4COLOR MENU C2
360,460,385,500,0
395,460,420,500,1
430,460,455,500,2
465,460,490,500,3
360,410,385,450,4
395,410,420,450,5
430,410,455,450,6
465,410,490,450,7
#,5,     C4CENTRAL MENU C2
5,430,45,480,7
55,430,95,480,7
105,430,145,480,7
155,430,195,480,7
14,425,2,"EDIT"
53,425,4,"CLUSTER"
110,425,6,"ROUTE"
162,425,3,"COVER"
70,502,7,"DIAL-A-RIDE PRINCIPAL SELECTION"
#,9,     C4EDIT MENU C2
5,430,45,480,7
55,430,95,430,7
105,430,145,480,7
155,430,195,480,7
205,430,245,480,7
6,425,2,"RESTORE"
58,425,4,"IGNORE"
58,414,4,"CLUSTR"
107,425,6,"IGNORE"
115,414,6,"ARC"
```

```
150,425,3,"SELECT"
150,414,3,"CLUSTR"
212,425,7,"EXIT"
70,502,7,"DIAL-A-RIDE CITY SUBFUNCTION"
#,12,   CACLUSTER MENU C2
5,430,40,430,7
50,430,85,480,7
95,430,130,480,7
140,430,175,480,7
185,430,220,480,7
230,430,265,480,7
275,430,310,480,7
5,425,6,"SELECT"
5,414,6,"CLUSTR"
50,425,3,"CREATE"
50,414,3,"CLUSTR"
95,425,2,"ADD TO"
95,414,2,"CLUSTR"
138,425,4,"DEL FRM"
142,414,4,"CLUSTR"
190,425,5,"RECST"
236,425,6,"XMIT"
279,425,7,"EXIT"
70,502,7,"DIAL-A-RIDE CLUSTER SUBFUNCTION"
#,5,   CAROUTING MENU C2
5,430,45,480,7
55,430,95,480,7
105,430,145,480,7
5,425,3,"SELECT"
5,414,3,"CLUSTR"
56,425,2,"ROUTE"
110,425,7,"EXIT"
```

```
70,502,7,"DIAL-A-RIDE ROUTING SUBFUNCTION"
#,5,    C4ROUTING SUB-MENU C2
355,430,390,480,7
395,430,410,480,7
415,430,440,480,7
445,430,470,480,7
475,430,500,480,7
355,414,2,"BGN"
395,414,6,"REV"
415,414,4,"END"
445,414,3,"REC"
475,414,5,"XIT"
#,3,     C4COVER MENU C2
5,430,45,480,7
55,430,95,480,7
10,425,2,"CHAIN"
61,425,4,"EXIT"
70,502,7,"DIAL-A-RIDE COVERING SELECTION"
```

```
#,      C3DIAL-A-RIDE SECONDARY MENU FILE C2
#,0,     C4COLOR MENU C2
360,460,385,500,0
395,460,420,500,1
430,460,455,500,2
465,460,490,500,3
360,410,385,450,4
395,410,420,450,5
430,410,455,450,6
465,410,490,450,7
#,3,     C4COVER MENU C2
5,430,45,480,7
55,430,95,480,7
10,425,2,"CHAIN"
41,425,4,"EXIT"
70,502,7,"DIAL-A-RIDE COVERING SELECTION"
#,10,    C4CHAINING MENU C2
5,430,40,480,7
50,430,85,480,7
95,430,130,480,7
140,430,175,480,7
195,430,220,480,7
5,425,6,"SELECT"
9,414,6,"CHAIN"
58,425,3,"EDIT"
50,414,3,"CHAINS"
103,425,2,"RCST"
95,414,2,"CHAINS"
150,425,4,"XPIT"
146,414,4,"CHAIN"
186,425,5,"FINALE"
```

```
70,502,7,"DIAL-A-RIDE CHAINING SUBFUNCTION"
4,12,    C4CHAIN EDIT FUNC C2
5,430,40,480,7
50,430,85,480,7
95,430,130,480,7
140,430,175,480,7
185,430,220,480,7
230,430,265,480,7
275,430,310,480,7
5,425,6,"SELECT"
9,414,6,"CHAIN"
50,425,5,"DISSOC"
95,425,4,"IGNORE"
99,414,4,"CHAIN"
140,425,3,"CREATE"
144,414,3,"CHAIN"
185,425,2,"ADD TO"
189,414,2,"CHAIN"
224,425,1,"RESTORE"
283,425,7,"EXIT"
70,502,7,"DIAL-A-RIDE CHAIN EDITTING SUBFUNCTION"
```

B-3

Cyber (FORTRAN)
Optimization Programs for
Dial-A-Ride

```
      PROGRAM [illegible](INPUT,OUTPUT)
      COMMON ITXOR(50),ITYOR(50),ITXDT(50),ITYDT(50),ITLOD(50),
     *       RTCOS(50),ICXOR(50),ICYOR(50),ICXDT(50),ICYDT(50),
     *       ICPTR(50),ICTRP(250),ISCOL(100,50),ISCOS(100),
     *       INTRP,INCLS,INCOL
      INTRP=INCLS=INCOL=0
C
C     INITIALIZE SET COVER COLUMNS
C
      DO 100 I=1,100
      DO 100 J=1,50
      ISCOL(I,J)=0
  100 CONTINUE
C
C     ENTER FUNCTION
C
  110 READ*, IFUNC
      IF(IFUNC.EQ.0) STOP
      IF(IFUNC.EQ.1) CALL TSTOR
      IF(IFUNC.EQ.2) CALL CLUST
      IF(IFUNC.EQ.3) CALL ROUTE
      IF(IFUNC.EQ.4) CALL CSTOR
      IF(IFUNC.EQ.5) CALL CHAIN
      IF(IFUNC.EQ.6) CALL SSTOR
      IF(IFUNC.EQ.7) CALL COVER
      GO TO 110
      END
      SUBROUTINE TSTOR
      COMMON ITXOR(50),ITYOR(50),ITXDT(50),ITYDT(50),ITLOD(50),
     *       RTCOS(50),ICXOR(50),ICYOR(50),ICXDT(50),ICYDT(50),
     *       ICPTR(50),ICTRP(250),ISCOL(100,50),ISCOS(100),
     *       INTRP,INCLS,INCOL
      DATA ITDIS/1/
C
C     FUNCTION 1 - TRIP STORAGE
C
C     ENTER # OF TRIPS & TRIP COORDINATES
C
      READ*, IREST,ITSTO
      IF(IREST.EQ.1) INTRP=INCOL=0
      DO 100 I=1,ITSTO
      READ*, IRXOR,IRYOR,IRXDT,IRYDT,IRLOD
C
C     STORE TRIPS
C
      ITXOR(INTRP+I)=IRXOR
      ITYOR(INTRP+I)=IRYOR
      ITXDT(INTRP+I)=IRXDT
      ITYDT(INTRP+I)=IRYDT
      ITLOD(INTRP+I)=IRLOD
      RTCOS(INTRP+I)=0.0
C
C     STORE COLUMNS
C
      ISCOL(INCOL+I,INTRP+I)=1
      IF(ITDIS.EQ.1) ISCOS(INCOL+I)=
     *    2*IFIX(SQRT(FLOAT((IRXOR-IRXDT)**2+(IRYOR-IRYDT)**2))+0.5)
      IF(ITDIS.EQ.2) ISCOS(INCOL+I)=
     *    2*((IRXOR-IRXDT)**2+(IRYOR-IRYDT)**2)
      IF(ITDIS.EQ.3) ISCOS(INCOL+I)=
     *    2*(IABS(IRXOR-IRXDT)+IABS(IRYOR-IRYDT))
  100 CONTINUE
      INTRP=INTRP+ITSTO
      INCOL=INCOL+ITSTO
```

```
      RETURN
      END
      SUBROUTINE CSTOR
      COMMON ITXOR(50),ITYOR(50),ITXDT(50),ITYDT(50),ITLOD(50),
     *       RTCOS(50),ICXOR(50),ICYOR(50),ICXDT(50),ICYDT(50),
     *       ICPTR(50),ICTRP(250),ISCOL(10,5),ISCOS(100),
     *       INTRP,INCLS,INCOL
      DIMENSION RTXOR(10),RTYOR(10),RTXDT(10),RTYDT(10),RCWGT(10)
      DATA RSTOLU/0.5/,RSTOLL/-0.5/,ITDIS/1/,RCTOL/0.1/
C
C     FUNCTION 4 - CLUSTER STORAGE
C
C     STORE CLUSTERS
C
      ITPTR=INCLS*0
      DO 130 I=1,INCOL
C
C     COMPUTE REDUCED COSTS FOR EACH COLUMN
C
      RRCOS=FLOAT(ISCOS(I))
      DO 100 J=1,INTRP
      IF(ISCOL(I,J).EQ.0) GO TO 100
      RRCOS=RRCOS-RTCOS(J)
  100 CONTINUE
      IF((RRCOS.GE.RSTOLU).OR.(RRCOS.LE.RSTOLL)) GO TO 130
      INCLS=INCLS+1
C
C     STORE CLUSTER TRIPS
C
      ITCOV=0
      DO 110 J=1,INTRP
      IF(ISCOL(I,J).EQ.0) GO TO 110
      ITCOV=ITCOV+1
      ICTRP(ITPTR+ITCOV)=J
      RTXOR(ITCOV)=FLOAT(ITXOR(J))
      RTYOR(ITCOV)=FLOAT(ITYOR(J))
      RTXDT(ITCOV)=FLOAT(ITXDT(J))
      RTYDT(ITCOV)=FLOAT(ITYDT(J))
      RCWGT(ITCOV)=FLOAT(ITLOD(J))
  110 CONTINUE
      ITPTR=ITPTR+ITCOV
      ICPTR(INCLS)=ITPTR
C
C     GENERATE & STORE CLUSTER COORDINATES
C
      RCXOR=FLOAT(ITXOR(ICTRP(ITPTR)))
      RCYOR=FLOAT(ITYOR(ICTRP(ITPTR)))
      RCXDT=FLOAT(ITXDT(ICTRP(ITPTR)))
      RCYDT=FLOAT(ITYDT(ICTRP(ITPTR)))
  120 CALL WSUM2(RCXOR,RCYOR,RCXDT,RCYDT,RTXOR,RTYOR,RTXDT,RTYDT,
     *           RCWGT,RWSOR,RWAOR,RWBOR,RWSDT,RWADT,RWBDT,ITCOV,ITDIS)
      RCGRA=SQRT((RWSOR*RCXOR-RWAOR)**2+(RWSOR*RCYOR-RWBOR)**2+
     *           (RWSDT*RCXDT-RWADT)**2+(RWSDT*RCYDT-RWBDT)**2)
      RCXOR=RWAOR/RWSOR
      RCYOR=RWBOR/RWSOR
      RCXDT=RWADT/RWSDT
      RCYDT=RWBDT/RWSDT
      IF(RCGRA.GT.RCTOL) GO TO 120
      ICXOR(INCLS)=IFIX(RCXOR+0.5)
      ICYOR(INCLS)=IFIX(RCYOR+0.5)
      ICXDT(INCLS)=IFIX(RCXDT+0.5)
      ICYDT(INCLS)=IFIX(RCYDT+0.5)
  130 CONTINUE
C
```

```
C       REINITIALIZE SET COVER COLUMNS
C
        DO 140 I=1,INCLS
        DO 140 J=1,INTRP
        ISCOL(I,J)=0
  140   CONTINUE
        INCOL=0
C
C       RETURN CLUSTERS
C
        ITPTR=0
        PRINT 160, INCLS
        DO 180 I=1,INCLS
        PRINT 150, ICXOR(I),ICYOR(I),ICXDT(I),ICYDT(I)
  150   FORMAT (4(I6,',',))
        IINCL=ICPTR(I)-ITPTR
        PRINT 160, IINCL
        DO 170 J=1,IINCL
        PRINT 160, ICTRP(ITPTR+J)
  160   FORMAT (I4,',',)
  170   CONTINUE
        ITPTR=ICPTR(I)
  100   CONTINUE
        RETURN
        END
        SUBROUTINE NSUM2(RCXOR,RCYOR,RCXDT,RCYDT,RTXOR,RTYOR,
     *                   RTXDT,RTYDT,RCWGT,RWSOR,RWAOR,RWBOR,
     *                   RWSDT,RWADT,RWBDT,ITCOV,ITDIS)
        DIMENSION RTXOR(10),RTYOR(10),RTXDT(10),RTYDT(10),RCWGT(10)
        DATA REPSL/0.01/
C
C       COMPUTE SUMS
C
        RWSOR=RWAOR=RWBOR=RWSDT=RWADT=RWBDT=0.0
        DO 100 I=1,ITCOV
        IF(RCWGT(I).LE.0.0) GO TO 100
        IF(ITDIS.EQ.1)
     * RDIST=SQRT((RCXOR-RTXOR(I))**2+(RCYOR-RTYOR(I))**2+REPSL)
        IF(ITDIS.EQ.2) RDIST=1.0
        IF(ITDIS.EQ.3)
     * RDIST=ABS(RCXOR-RTXOR(I))+ABS(RCYOR-RTYOR(I))+REPSL
        RWSOR=RWSOR+RCWGT(I)/RDIST
        RWAOR=RWAOR+RCWGT(I)*RTXOR(I)/RDIST
        RWBOR=RWBOR+RCWGT(I)*RTYOR(I)/RDIST
        IF(ITDIS.EQ.1)
     * RDIST=SQRT((RCXDT-RTXDT(I))**2+(RCYDT-RTYDT(I))**2+REPSL)
        IF(ITDIS.EQ.2) RDIST=1.0
        IF(ITDIS.EQ.3)
     * RDIST=ABS(RCXDT-RTXDT(I))+ABS(RCYDT-RTYDT(I))+REPSL
        RWSDT=RWSDT+RCWGT(I)/RDIST
        RWADT=RWADT+RCWGT(I)*RTXDT(I)/RDIST
        RWBDT=RWBDT+RCWGT(I)*RTYDT(I)/RDIST
  100   CONTINUE
        RETURN
        END
        SUBROUTINE SSTOR
        COMMON ITXOR(50),ITYOR(50),ITXDT(50),ITYDT(50),ITLOD(50),
     *         RTCOS(50),ICXOR(50),ICYOR(50),ICXDT(50),ICYDT(50),
     *         ICPTR(50),ICTRP(250),ISCOL(100,50),ISCOS(100),
     *         INTRP,INCLS,INCOL
C
C       FUNCTION 6 - COLUMN STORAGE
C
C       ENTER # OF COLUMNS
```

```
C
      READ*, ICSTO
      DO 110 I=1,ICSTO
C
C     ENTER COLUMN PARAMETERS & TRIP NUMBERS
C
      READ*, IRCOS,ITCOV
C
C     STORE COLUMN
C
      DO 100 J=1,ITCOV
      READ*, ITNUM
      ISCOL(INCOL+I,ITNUM)=1
  100 CONTINUE
      ISCOS(INCOL+I)=IRCOS
  110 CONTINUE
      INCOL=INCOL+ICSTO
      RETURN
      END
      SUBROUTINE CLUST
      COMMON ITXOR(50),ITYOR(50),ITXDT(50),ITYDT(50),ITLOD(50),
     *       RTCOS(50),ICXOR(50),ICYOR(50),ICXDT(50),ICYDT(50),
     *       ICPTR(50),ICTPR(250),ISCOL(100,50),ISCOS(100),
     *       INTRP,INCLS,INCOL
      LOGICAL ZPATH
      DIMENSION ICAPT(50),ILOAD(50),IDIST(50,50),RCOST(50),
     *          IASST(50),ICUAL(100),IUPPT(100),IDNPT(100),
     *          IFTPT(100),ZPATH(100),RCXOR(50),RCYOR(50),
     *          RCXDT(50),RCYDT(50),RTXOR(50),RTYOR(50),
     *          RTXDT(50),RTYDT(50),ITNUM(50),IPARM(7)
      DATA ITDIS/1/,RDISM/1.0/
C
C     FUNCTION 2 - CLUSTERING
C
C     ENTER CLUSTERING PARAMETERS & CLUSTER COORDINATES
C
      READ*, ILOCM,IRCLS,IRTRP,IVCAP,IDCAP
      IPARM(3)=IRCLS
      IF(IDCAP.GT.0) IPARM(3)=IPARM(3)+1
      IPARM(4)=IRTRP+1
      ITOTC=0
      DO 110 I=1,IRCLS
      RCXOR(I)=RCYOR(I)=RCXDT(I)=RCYDT(I)=0.0
      IF(ILOCM.NE.1) GO TO 100
      READ*, RCXOR(I),RCYOR(I),RCXDT(I),RCYDT(I)
  100 ICAPT(I)=IVCAP
      ITOTC=ITOTC+ICAPT(I)
  110 CONTINUE
      ICAPT(IRCLS+1)=IDCAP
      ITOTC=ITOTC+IDCAP
C
C     ENTER TRIPS
C
      ITOTL=0
      DO 120 I=1,IRTRP
      READ*, IRNUM
      ITNUM(I)=IRNUM
      RTXOR(I)=FLOAT(ITXOR(IRNUM))
      RTYOR(I)=FLOAT(ITYOR(IRNUM))
      RTXDT(I)=FLOAT(ITXDT(IRNUM))
      RTYDT(I)=FLOAT(ITYDT(IRNUM))
      ILOAD(I)=ITLOD(IRNUM)
      RCOST(I)=RTCOS(IRNUM)
      ITOTL=ITOTL+ILOAD(I)
```

```
 120 CONTINUE
     ILOAD(IRTR+1)=ITCTO-ITCTL
     IPARM(7)=0
C
C    GENERATE CLUSTERS
C
 130 CALL DIST(IDIST,RCOST,RCXOR,RCYOR,RCXDT,RCYDT,RTXOR,RTYOR,
    *           RTXDT,RTYDT,IPARM,IDCAP,ITDIS,RDTSM)
     CALL TF(50,50,100,IPARM,IDNPT,ILOAD,IDIST,JVSET,IDUAL,
    *        IDPAT,IDNPT,INTPT,ZMATH)
     IF(ILOCN.EQ.1) GO TO 140
     CALL LOCAT(RCXOR,RCYOR,RCXDT,RCYDT,RTXOR,RTYOR,RTXDT,RTYDT,
    *           IDNPT,IASST,IPARM,IDCAP,ITDIS,ISTOP)
     IPARM(7)=1
     IF(ISTOP.EQ.0) GO TO 130
C
C    RETURN CLUSTER ASSIGNMENTS
C
 140 IF(IRCLS.EQ.1) GO TO 190
     DO 180 I=2,IRCLS
     IF(IASST(I).EQ.0) GO TO 180
     J=IDNPT(I)-IPARM(3)
     IF(J.GT.IRTRP) GO TO 180
     PRINT 170, I,ITNUM(J)
 170 FORMAT (2(I4,*,*))
 180 CONTINUE
 190 IINCL=IPARM(3)+1
     IINCU=IPARM(3)+IRTRP
     DO 200 J=IINCL,IINCU
     IF(IASST(J).EQ.0) GO TO 200
     I=IDNPT(J)
     IF(I.GT.IRCLS) GO TO 200
     PRINT 170, I,ITNUM(J-IPARM(3))
 200 CONTINUE
     RETURN
     END
     SUBROUTINE DIST(IDIST,RCOST,RCXOR,RCYOR,RCXDT,RCYDT,RTXOR,RTYOR,
    *                RTXDT,RTYDT,IPARM,IDCAP,ITDIS,RDTSM)
     DIMENSION IDIST(50,50),RCXOR(50),RCYOR(50),RCXDT(50),RCYDT(50),
    *          RTXOR(50),RTYOR(50),RTXDT(50),RTYDT(50),RCOST(50),
    *          IPARM(7)
     IRCLS=IPARM(3)
     IF(IDCAP.GT.0) IRCLS=IRCLS-1
     IRTRP=IPARM(4)+1
C
C    COMPUTE DISTANCES
C
     RTRCL=FLOAT(IRTRP)/FLOAT(IRCLS)
     DO 110 I=1,IRCLS
     RDXOR=RCXOR(I)
     RDYOR=RCYOR(I)
     RDXDT=RCXDT(I)
     RDYDT=RCYDT(I)
     IF(RCXOR.GT.0.0.AND.RDYOR.GT.0.0.AND.
    *   RDXDT.GT.0.0.AND.RDYDT.GT.0.0) GO TO 100
     J=IFIX(RTRCL*FLOAT(I))
     RDXOR=RTXOR(J)
     RDYOR=RTYOR(J)
     RDXDT=RTYDT(J)
     RDYDT=RTYDT(J)
 100 DO 105 J=1,IRTRP
     IF(ITDIS.EQ.1)
    * RDIST=SQRT((RDXOR-RTXOR(J))**2+
    *            (RDYOR-RTYOR(J))**2)+
```

```
     *          SQRT((ROXDT-RTXDT(J))**2+
     *               (ROYDT-RTYDT(J))**2)
      IF(ITDIS.EQ.2)
     + RDIST=(ROXOR-RTXOR(J))**2+(ROYOR-RTYOR(J))**2+
     *          (ROXDT-RTXDT(J))**2+(ROYDT-RTYDT(J))**2
      IF(ITDIS.EQ.3)
     + RDIST=ABS(ROXOR-RTXOR(J))+ABS(ROYOR-RTYOR(J))+
     *          ABS(ROXDT-RTXDT(J))+ABS(ROYDT-RTYDT(J))
      IDIST(I,J)=IFIX(RDISM*RDIST+LCOST(J)+0.5)
  105 CONTINUE
      IDIST(I,IRTRP+1)=0
  110 CONTINUE
      IF(IDCAP.EQ.0) RETURN
      IINDU=IRTRP+1
      DO 115 J=1,IINDU
      IDIST(IRCLS+1,J)=0
  115 CONTINUE
      RETURN
      END
      SUBROUTINE LOCAT(RCXOR,RCYOR,RCXDT,RCYDT,RTXOR,RTYOR,
     *                 RTXDT,RTYDT,IDNPT,IASST,IPARM,IDCAP,
     *                 ITDIS,ISTOP)
      DIMENSION RCXOR(50),RCYOR(50),RCXDT(50),RCYDT(50),
     *          RTXOR(50),RTYOR(50),RTXDT(50),RTYDT(50),
     *          IDNPT(100),IASST(100),RCWGT(50,50),
     *          RWSOR(50),RWAOR(50),RWBOR(50),RWSDT(50),
     *          RWADT(50),RWBDT(50),IPARM(7)
      DATA RTOL1/0.1/
      IRCLS=IPARM(3)
      IRTRP=IPARM(4)
      RTOL2=SQRT(RTOL1**2/FLOAT(IRCLS))
C
C     GENERATE CLUSTER WEIGHTS
C
      DO 100 I=1,IRCLS
      DO 100 J=1,IRTRP
      RCWGT(I,J)=0.0
  100 CONTINUE
      IF(IRCLS.EQ.1) GO TO 110
      DO 105 I=2,IRCLS
      RCWGT(I,IDNPT(I)-IRCLS)=FLOAT(IASST(I))
  105 CONTINUE
  110 IINDL=IRCLS+1
      IINDU=IRCLS+IRTRP
      DO 115 J=IINDL,IINDU
      RCWGT(IDNPT(J),J-IRCLS)=FLOAT(IASST(J))
  115 CONTINUE
C
C     COMPUTE GRADIENT
C
      RTGRA=0.0
      IINDL=IRCLS
      IF(IDCAP.GT.0) IINDU=IINDL-1
      DO 120 I=1,IINDU
      CALL WSUM1(I,RCXOR(I),RCYOR(I),RCXDT(I),RCYDT(I),RTXOR,
     *           RTYOR,RTXDT,RTYDT,RCWGT,RWSOR(I),RWAOR(I),
     *           RWBOR(I),RWSDT(I),RWADT(I),RWBDT(I),IPARM,ITDIS)
      RTGRA=RTGRA+(RWSOR(I)*RCXOR(I)-RWAOR(I))**2+
     *            (RWSOR(I)*RCYOR(I)-RWBOR(I))**2+
     *            (RWSDT(I)*RCXDT(I)-RWADT(I))**2+
     *            (RWSDT(I)*RCYDT(I)-RWBDT(I))**2
  120 CONTINUE
      RTGRA=SQRT(RTGRA)
      IF(RTGRA.GT.RTOL1) GO TO 130
```

```
      ISTOP=1
      RETURN
C
C     UPDATE CLUSTER POINTS
C
  130 DO 150 I=1,IINDU
      IF(RWSOR(I).GT.0.0) GO TO 140
      ROXOR(I)=ROYOR(I)=ROXDT(I)=ROYDT(I)=0.0
      GO TO 150
  140 ROXOR(I)=RWAOR(I)/RWSOR(I)
      ROYOR(I)=RWBOR(I)/RWSOR(I)
      ROXDT(I)=RWADT(I)/RWSDT(I)
      ROYDT(I)=RWBDT(I)/RWSDT(I)
      CALL WSUM1(I,ROXOR(I),ROYOR(I),ROXDT(I),ROYDT(I),RTXOR,
     *           RTYOR,RTXDT,RTYDT,RCWGT,RWSOR(I),RWAOR(I),
     *           RWBOR(I),RWSDT(I),RWADT(I),RWBDT(I),IPARM,ITDIS)
      RPGRA=SQRT((RWSOR(I)*ROXOR(I)-RWAOR(I))**2+
     *           (RWSOR(I)*ROYOR(I)-RWBOR(I))**2+
     *           (RWSDT(I)*ROXDT(I)-RWADT(I))**2+
     *           (RWSDT(I)*ROYDT(I)-RWBDT(I))**2)
      IF(RPGRA.GT.RTOL2) GO TO 145
  150 CONTINUE
      ISTOP=0
      RETURN
      END
      SUBROUTINE WSUM1(I,ROXOR,ROYOR,ROXDT,ROYDT,RTXOR,RTYOR,
     *                 RTXDT,RTYDT,RCWGT,RWSOR,RWAOR,RWBOR,
     *                 RWSDT,RWADT,RWBDT,IPARM,ITDIS)
      DIMENSION RTXOR(50),RTYOR(50),RTXDT(50),RTYDT(50),RCWGT(50,50),
     *          IPARM(7)
      DATA REPSL/0.01/
      IRTRP=IPARM(4)-1
C
C     COMPUTE SUMS
C
      RWSOR=RWAOR=RWBOR=RWSDT=RWADT=RWBDT=0.0
      DO 100 J=1,IRTRP
      IF(RCWGT(I,J).LE.0.0) GO TO 100
      IF(ITDIS.EQ.1)
     * RDIST=SQRT((ROXOR-RTXOR(J))**2+(ROYOR-RTYOR(J))**2+REPSL)
      IF(ITDIS.EQ.2) RDIST=1.0
      IF(ITDIS.EQ.3)
     * RDIST=ABS(ROXOR-RTXOR(J))+ABS(ROYOR-RTYOR(J))+REPSL
      RWSOR=RWSOR+RCWGT(I,J)/RDIST
      RWAOR=RWAOR+RCWGT(I,J)*RTXOR(J)/RDIST
      RWBOR=RWBOR+RCWGT(I,J)*RTYOR(J)/RDIST
      IF(ITDIS.EQ.1)
     * RDIST=SQRT((ROXDT-RTXDT(J))**2+(ROYDT-RTYDT(J))**2+REPSL)
      IF(ITDIS.EQ.2) RDIST=1.0
      IF(ITDIS.EQ.3)
     * RDIST=ABS(ROXDT-RTXDT(J))+ABS(ROYDT-RTYDT(J))+REPSL
      RWSDT=RWSDT+RCWGT(I,J)/RDIST
      RWADT=RWADT+RCWGT(I,J)*RTXDT(J)/RDIST
      RWBDT=RWBDT+RCWGT(I,J)*RTYDT(J)/RDIST
  100 CONTINUE
      RETURN
      END
      SUBROUTINE ROUTE
      COMMON ITXOR(50),ITYOR(50),ITXDT(50),ITYDT(50),ITLOD(50),
     *       RTCOS(50),ICXOR(50),ICYOR(50),ICXDT(50),ICYDT(50),
     *       ICPTR(50),ICTOP(250),ISCOL(100,50),ISCOS(100),
     *       INTRP,INCLS,INCOL
      DIMENSION IDCOS(50,50),ITNUM(50),ISTOP(50)
      DATA ISLOW/999999/,ITDIS/1/
```

```
C
C     FUNCTION 3 - ROUTING
C
C     ENTER ROUTING PARAMETERS & TRIP NUMBERS IN ROUTE
C
      READ*, IRTRP,IINUM,IFNUM,ITCTP
      IIIND=IFIND=0
      DO 100 I=1,IRTRP
      READ*, IRNUM
      ITNUM(I)=ITNUM(I+IRTRP)=IRNUM
      IF(IRNUM.EQ.IINUM) IIIND=I
      IF(IRNUM.EQ.IFNUM) IFIND=I+IRTRP
  100 CONTINUE
C
C     GENERATE COST MATRIX
C
      DO 110 I=1,IRTRP
      DO 110 J=1,IRTRP
      IICOS(I,J)=IBIGM
      IF(I.NE.J)
     * IICOS(I,J)=IDIST(ITDIS,ITXOR(ITNUM(I)),ITYOR(ITNUM(I)),
     *                            ITXOR(ITNUM(J)),ITYOR(ITNUM(J)))
      IICOS(I,J+IRTRP)=IDIST(ITDIS,ITXOR(ITNUM(I)),ITYOR(ITNUM(I)),
     *                                 ITXDT(ITNUM(J)),ITYDT(ITNUM(J)))
      IICOS(I+IRTRP,J)=IBIGM
      IICOS(I+IRTRP,J+IRTRP)=IBIGM
      IF((I+IRTRP).NE.(J+IRTRP))
     * IICOS(I+IRTRP,J+IRTRP)=IDIST(ITDIS,ITXDT(ITNUM(I)),
     *                                         ITYDT(ITNUM(I)),
     *                                         ITXDT(ITNUM(J)),
     *                                         ITYDT(ITNUM(J)))
  110 CONTINUE
      IINDU=2*IRTRP
      DO 120 I=1,IINDU
      IICOS(IINDU+1,I)=IBIGM
      IF(I.EQ.IIIND) IICOS(IINDU+1,I)=0
      IF((I.LE.IRTRP).AND.(IIIND.EQ.0)) IICOS(IINDU+1,I)=0
      IICOS(I,IINDU+1)=IBIGM
      IF(I.EQ.IFIND) IICOS(I,IINDU+1)=0
      IF((I.GT.IRTRP).AND.(IFIND.EQ.0)) IICOS(I,IINDU+1)=0
  120 CONTINUE
      IICOS(IINDU+1,IINDU+1)=IBIGM
      INNOD=2*IRTRP+1
C
C     GENERATE ROUTE
C
      CALL SOLVE(INNOD,IICOS,ICCOS,ICTOR)
C
C     RETURN ROUTE DISTANCE & SEQUENCES
C
      IF(ITOTP.EQ.2) GO TO 175
      DO 130 J=1,INNOD
      IF(ICTOR(J).EQ.INNOD) GO TO 140
  130 CONTINUE
  140 DO 160 I=1,IRTRP
      J=J+1
      IF(J.GT.INNOD) J=1
      PRINT 150, ITNUM(ICTOR(J))
  150 FORMAT (I5,*,*)
  160 CONTINUE
      DO 170 I=1,IRTRP
      J=J+1
      IF(J.GT.INNOD) J=1
      PRINT 150, -ITNUM(ICTOR(J))
```

```
  170 CONTINUE
C
C     STORE COLUMN
C
  175 IF(ITOTR.EQ.1) RETURN
      DO 180 I=1,IRTRP
      ISCOL(INCOL+1,ITRUM(I))=1
  180 CONTINUE
      ISCOS(INCOL+1)=IOCOS
      INCOL=INCOL+1
      RETURN
      END
      FUNCTION IDIST(ITDIS,IDXOR,IDYOR,IDXOT,IDYOT)
      IF(ITDIS.EQ.1)
     * IDIST=IFIX(SQRT(FLOAT((IDXOR-IDXOT)**2+(IDYOR-IDYOT)**2))+0.5)
      IF(ITDIS.EQ.2)
     * IDIST=(IDXOR-IDXOT)**2+(IDYOR-IDYOT)**2
      IF(ITDIS.EQ.3)
     * IDIST=IABS(IDXOR-IDXOT)+IABS(IDYOR-IDYOT)
      RETURN
      END
      SUBROUTINE SOLVE(INNOD,IICOS,IOCOS,IOTOR)
C
C     THIS SUBROUTINE SOLVES THE PROBLEM USING
C     A LEAST-LOWER-BOUND BRANCHING RULE
C
      DIMENSION IICOS(50,50),ISTOR(2750),IOTOR(50),INVIS(50),
     *          ISPTR(275),IASST(50),ILAMD(50),IIHAT(50),
     *          ICHAT(50),ILIST(50),IMARK(50),IZLIS(100),
     *          IREST(12)
      DATA IBIGN/999999/,IMSUB/275/,IMSTO/2750/,ICMAX/19999/,
     *     IMASS/100/
      RTIME=SECOND(CP)
      INASS=INSUB=IBPTR=0
      IIPTR=1
C
C     INITIALIZE SUBTOUR LIST
C
      ICPTR=1
      IINCU=IMSTO/(INNOD+5)-1
      DO 100 I=1,IINCU
      INPTR=ICPTR+INNOD+5
      ISTOR(ICPTR)=INPTR
      ICPTR=INPTR
  100 CONTINUE
      ISTOR(ICPTR)=0
      IOCOS=INNOD*ICMAX+1
      IREST(11)=0
      IREST(12)=IOCOS
C
C     SOLVE ASSIGNMENT PROBLEM
C
  110 CALL HUNGARY(50,100,INNOD,IICOS,IASST,ILAMD,IIHAT,ICHAT,
     *             ILIST,IMARK,IZLIS,IREST)
      IMCOS=IREST(9)+IREST(10)
      INASS=INASS+1
      IF(INASS.GE.IMASS) GO TO 400
  115 IF(IMCOS.GE.IOCOS) GO TO 270
C
C     FIND MINIMUM CARDINALITY SUBTOUR
C
      IMCAR1=IMCAR2=INNOD
      IMENT=1
      DO 120 I=1,INNOD
```

```
      INVIS(I)=0
  120 CONTINUE
  130 DO 140 I=1,INNOD
      IF(INVIS(I).EQ.0) GO TO 150
  140 CONTINUE
      GO TO 180
  150 INENT=ICENT=I
      ICCAR1=0
      DO 160 I=1,INNOD
      IPENT=INENT
      INENT=IASST(INENT)
      IF(INVIS(INENT).EQ.1) GO TO 170
      INVIS(INENT)=1
      IF(IICOS(IPENT,INENT).LT.0) ICCAR1=ICCAR1-1
  160 CONTINUE
  170 ICCAR1=ICCAR1+I-1
      ICCAR2=I-1
      IF(ICCAR1.GE.IMCAR1) GO TO 130
      IMCAR1=ICCAR1
      IMCAR2=ICCAR2
      IMENT=ICENT
      GO TO 130
  180 IF(IMCAR2.LT.INNOD) GO TO 240
C
C     SAVE MINIMUM COST TOUR
C
      IOCCS=IMCCS
      INENT=IMENT
      DO 190 I=1,INNOD
      INENT=IASST(INENT)
      IOTOR(I)=INENT
  190 CONTINUE
      IF(INSUB.EQ.0) GO TO 270
C
C     REMOVE FATHOMED SUBPROBLEMS
C
      DO 230 I=1,INSUB
  200 IF(ISTOR(ISPTR(I)+2).LT.IOCCS) GO TO 230
      IRPTR=ISPTR(I)
  210 IPPTR=ISTOR(IRPTR)
      ISTOR(IRPTR)=IIPTR
      IIPTR=IRPTR
      IF(IPPTR.EQ.0) GO TO 220
      ISTOR(IPPTR+3)=ISTOR(IPPTR+3)-1
      IF(ISTOR(IPPTR+3).GT.0.OR.IPPTR.EQ.IRPTR) GO TO 220
      IRPTR=IPPTR
      GO TO 210
  220 ISPTR(I)=ISPTR(INSUB)
      INSUB=INSUB-1
      IF(I.LE.INSUB) GO TO 200
  230 CONTINUE
      GO TO 270
C
C     ADD SUBTOUR TO LIST
C
  240 IF(IIPTR.GT.0) GO TO 250
      GO TO 400
  250 IAPTR=IIPTR
      IIPTR=ISTOR(IIPTR)
      IF(INSUB.LT.IMSUB) GO TO 260
      GO TO 400
  260 INSUB=INSUB+1
      ISPTR(INSUB)=IAPTR
      ISTOR(IAPTR)=IBPTR
```

```
      ISTOR(IBPTR+1)=ISNOD
      ISTOR(IBPTR+2)=INCOS
      ISTOR(IBPTR+3)=1
      ISTOR(IBPTR+4)=INCAR2
      INENT=IMENT
      DO 265 I=1,INCAR2
      IPENT=INENT
      INENT=IABS(INENT)
      IF(IICOS(IPENT,INENT).LT.0) IPENT=-IPENT
      ISTOR(IBPTR+I+4)=IPENT
  265 CONTINUE
      IF(IBPTR.EQ.0) GO TO 315
      ISTOR(IBPTR+3)=ISTOR(IBPTR+3)+1
C
C     RESTORE ARCS
C
  270 IF(IBPTR.EQ.0) GO TO 315
      IIROW=ISTOR(IBPTR+ISNOD+4)
      IICOL=IABS(ISTOR(IBPTR+ITNOD+4))
      IICOS(IIROW,IICOL)=IICOS(IIROW,IICOL)-IBIGM
      IF((ILAMD(IIROW)+ICHAT(IICOL)).LE.IICOS(IIROW,IICOL))
     *     GO TO 275
      IREST(9)=IREST(9)+IICOS(IIROW,IICOL)
     *                 -ILAMD(IIROW)-ICHAT(IICOL)
      ILAMD(IIROW)=IICOS(IIROW,IICOL)-ICHAT(IICOL)
  275 IPNOD=ISNOD
      ISNOD=ISNOD+1
      IF(ISNOD.LE.ISCAR) GO TO 360
      IINDU=ISCAR-1
      DO 280 I=1,IINDU
      IIROW=ISTOR(IBPTR+I+4)
      IF(IIROW.LT.0) GO TO 280
      IICOL=IABS(ISTOR(IBPTR+I+5))
      IICOS(IIROW,IICOL)=IICOS(IIROW,IICOL)-IBIGM
      IREST(9)=IREST(9)-IBIGM
  280 CONTINUE
      ISNOD=ISTOR(IBPTR+1)
      IBPTR=ISTOR(IBPTR)
  285 IF(IBPTR.EQ.0) GO TO 305
      ISCAR=ISTOR(IBPTR+4)
      IIROW=ISTOR(IBPTR+ISNOD+4)
      ITNOD=ISNOD+1
      IF(ITNOD.GT.ISCAR) ITNOD=1
      IICOL=IABS(ISTOR(IBPTR+ITNOD+4))
      IICOS(IIROW,IICOL)=IICOS(IIROW,IICOL)-IBIGM
      IF((ILAMD(IIROW)+ICHAT(IICOL)).LE.IICOS(IIROW,IICOL))
     *     GO TO 290
      IREST(9)=IREST(9)+IICOS(IIROW,IICOL)
     *                 -ILAMD(IIROW)-ICHAT(IICOL)
      ILAMD(IIROW)=IICOS(IIROW,IICOL)-ICHAT(IICOL)
  290 IINDU=ISNOD-1
      IF(IINDU.LT.1) GO TO 300
      DO 295 I=1,IINDU
      IIROW=ISTOR(IBPTR+I+4)
      IF(IIROW.LT.0) GO TO 295
      IICOL=IABS(ISTOR(IBPTR+I+5))
      IICOS(IIROW,IICOL)=IICOS(IIROW,IICOL)+IBIGM
      IREST(9)=IREST(9)-IBIGM
  295 CONTINUE
  300 ISNOD=ISTOR(IBPTR+1)
      IBPTR=ISTOR(IBPTR)
      GO TO 285
C
C     REMOVE UNBRANCHED PROBLEM
```

```
C
  305 IPPTR=IBPTR
  310 IF(ISTOR(IPPTR+3).GT.0) GO TO 315
      IIPTR=IPPTR
      IPPTR=ISTOR(IPPTR)
      ISTOR(IIPTR)=IKPTR
      IKPTR=IKPTR
      IF(IPPTR.EQ.0) GO TO 315
      ISTOR(IPPTR+3)=ISTOR(IPPTR+3)-1
      GO TO 310
C
C     BRANCH ON LEAST LOWER BOUND SUBPROBLEM
C
  315 IF(INSUB.EQ.0) GO TO 400
      IBPTR=0
      IBCCS=IBIGM
      DO 320 I=1,INSUB
      IF(ISTOR(ISPTR(I)+2).GE.IBCCS) GO TO 320
      IBPTR=ISPTR(I)
      IBIND=I
      IBCCS=ISTOR(IBPTR+2)
  320 CONTINUE
      IF(IBPTR.EQ.0) GO TO 400
      ISPTR(IBIND)=ISPTR(INSUB)
      INSUB=INSUB-1
C
C     PROHIBIT AND REQUIRE ARCS
C
      ISNOD=ISTOR(IBPTR+1)
      IPPTR=ISTOR(IBPTR)
  330 IF(IPPTR.EQ.0) GO TO 350
      ISCAR=ISTOR(IPPTR+4)
      IIROW=ISTOR(IPPTR+ISNOD+4)
      ITNOD=ISNOD+1
      IF(ITNOD.GT.ISCAR) ITNOD=1
      IICOL=IABS(ISTOR(IPPTR+ITNOD+4))
      IICOS(IIROW,IICOL)=IICOS(IIROW,IICOL)+IBIGM
      IINDU=ISNOD-1
      IF(IINDU.LT.1) GO TO 345
      DO 340 I=1,IINDU
      IIROW=ISTOR(IPPTR+I+4)
      IF(IIROW.LT.0) GO TO 340
      IICOL=IABS(ISTOR(IPPTR+I+5))
      IICOS(IIROW,IICOL)=IICOS(IIROW,IICOL)-IBIGM
      IREST(9)=IREST(9)+IBIGM
      IF((ILAMD(IIROW)+ICHAT(IICOL)).LE.IICOS(IIROW,IICOL))
     *      GO TO 340
      IREST(9)=IREST(9)+IICOS(IIROW,IICOL)
     *           -ILAMD(IIROW)-ICHAT(IICOL)
      ILAMD(IIROW)=IICOS(IIROW,IICOL)-ICHAT(IICOL)
  340 CONTINUE
  345 ISNOD=ISTOR(IPPTR+1)
      IPPTR=ISTOR(IPPTR)
      GO TO 330
  350 ISNOD=1
      IPNOD=0
      ISCAR=ISTOR(IBPTR+4)
  360 IIROW=ISTOR(IBPTR+ISNOD+4)
      IF(IIROW.GT.0) GO TO 365
      ISNOD=ISNOD+1
      GO TO 360
  365 ITNOD=ISNOD+1
      IF(ITNOD.GT.ISCAR) ITNOD=1
      IICOL=IABS(ISTOR(IBPTR+ITNOD+4))
```

```
      IICOS(IIROW,IICOL)=IICOS(IIROW,IICOL)+IHIGH
      IF(IRNOD.EQ.0) GO TO 375
      IIROW=ISTOP([illegible]INOD-1)
      IF(IIROW.LT.0) GO TO 370
      IICOL=INUS(I[illegible](ISPTR+IRNOD+5))
      IICOS(IIROW,IICOL)=IICOS(IIROW,IICOL)-IHIGH
      IREST(9)=IREST(9)-IHIGH
      IF((ILAND(IIROW)+ICHAT(IICOL)).NE.IICOS(IIROW,IICOL))
     *     GO TO 370
      IREST(9)=IREST(9)+IICOS(IIROW,IICOL)
     *                  -ILAND(IIROW)-ICHAT(IICOL)
      ILAND(IIROW)=IICOS(IIROW,IICOL)-ICHAT(IICOL)
  370 CONTINUE
  375 IREST(11)=1
      IREST(12)=ICCOS
      IF(IREST(9).LT.ICCOS) GO TO 110
      ICCOS=IREST(9)
      GO TO 115
C
C     RETURN
C
  400 RETURN
      END
      SUBROUTINE CHAIN
      COMMON ITXOR(50),ITYOR(50),ITXDT(50),ITYDT(50),ITLOD(50),
     *        RTCOS(50),OX(50),OY(50),DX(50),DY(50),
     *        ICPTR(50),ICTRP(250),ISCOL(100,50),ISCOS(100),
     *        INTRP,M,INCOL
      COMMON/NET/F,NAM,NS,NT,LC,HI,COST
      DIMENSION ITRIPC(100)
      DIMENSION NAM(100),NS(100),NT(100),LC(100),HI(100),F(100)
      DIMENSION COST(100),NCLS(100),NUSE(100)
      EQUIVALENCE (NAM(1),G(1,1)),(NS(1),G(1,2)),(NT(1),G(1,3))
      EQUIVALENCE (LC(1),G(1,4)),(HI(1),G(1,5)),(COST(1),G(1,6))
      INTEGER C(20)
      INTEGER PATH(500),NPATH(500)
      INTEGER F
      INTEGER K(20),G(100,5),S(50,50)
      INTEGER CX,OY,DX,DY
C
C     ENTERING CHAINING PARAMETER
C
      READ*, ICMAX, INCLS
C
C     READ CLUSTERS
C
      READ*,(NCLS(I),I=1,INCLS)
      DO 450 I=1,M
450   NUSE(I)=0
      DO 460 I=1,INCLS
460   NUSE(NCLS(I))=1
C
C     GENERATING DISTANCE MATRIX
C
      M1=M+1
      M2=2*M
      L=1
      DO 30 I=1,M
      DO 20 J=1,M
      S(I,J)=((DX(I)-OX(J))**2+(DY(I)-OY(J))**2)**(.5)+.5
20    CONTINUE
30    CONTINUE
C
C         DEFINING THE 3 ORIGINS CLOSEST TO EACH DESTINATION
```

```
C
      L=1
      DO 900 I=1,M
      K(1)=999
      K(2)=999
      K(3)=999
      DO 901 J=1,M
      IF (J.EQ.I) GOTO 901
      IF (S(I,J).GT.K(3)) GOTO 901
      IF (S(I,J).GT.K(2)) GOTO 140
      IF (S(I,J).GT.K(1)) GOTO 110
      K(3)=K(2)
      K(2)=K(1)
      O(3)=O(2)
      O(2)=O(1)
      K(1)=S(I,J)
      O(1)=J
      GOTO 901
110   K(3)=K(2)
      O(3)=O(2)
      K(2)=S(I,J)
      O(2)=J
      GOTO 901
140   K(3)=S(I,J)
      O(3)=J
901   CONTINUE
      DO 902 J=1,3
      G(L,1)=L
      G(L,2)=I+M
      G(L,3)=O(J)
      G(L,4)=0
      G(L,5)=1
      G(L,6)=K(J)
      L=L+1
902   CONTINUE
900   CONTINUE
      M3=3*M
C
C            DEFINING THE THREE DESTINATION POINTS CLOSEST TO EACH ORIGIN
C
      L=1
      DO 905 J=1,M
      K(1)=999
      K(2)=999
      K(3)=999
      DO 906 I=1,M
      IF (J.EQ.I)GOTO 906
      IF (S(I,J).GT.K(3))GOTO 906
      IF (S(I,J).GT.K(2))GOTO 240
      IF (S(I,J).GT.K(1))GOTO 210
      K(3)=K(2)
      K(2)=K(1)
      O(3)=O(2)
      O(2)=O(1)
      O(1)=I
      K(1)=S(I,J)
      GOTO 906
210   K(3)=K(2)
      O(3)=O(2)
      O(2)=I
      K(2)=S(I,J)
      GOTO 906
240   K(3)=S(I,J)
      O(3)=I
```

```
 906    CONTINUE
        DO 905 M=1,J
        DO 907 I=1,[illegible]
        IF ([illegible]) GO TO 906
 907    CONTINUE
 250    G(M*2+L,1)=M*2+L
        G(M*2+L,2)=C(L)+N
        G(M*3+L,3)=J
        G(M*3+L,4)=[illegible]
        G(M*3+L,5)=1
        G(M*3+L,6)=X(L)
        L=L+1
 906    CONTINUE
 905    CONTINUE
C
C               CREATING A SUPER SOURCE
C
        DO 911 I=1,M
        G(3*M+L+I-1,1)=3*M+L+I-1
        G(3*M+L+I-1,2)=2*M+1
        G(3*M+L+I-1,3)=I
        G(3*M+L+I-1,4)=0
        G(3*M+L+I-1,5)=1
        G(3*M+L+I-1,6)=STOT
 911    CONTINUE
C
C             CREATING A SUPER SINK
C
 921    DO 912 I=M1,M2
        G(3*M+L+I-1,1)=3*M+L+I-1
        G(3*M+L+I-1,2)=I
        G(3*M+L+I-1,3)=2*M+2
        G(3*M+L+I-1,4)=0
        G(3*M+L+I-1,5)=1
        G(3*M+L+I-1,6)=STOT
 912    CONTINUE
        M7=5*M+L
        M5=2*M+1
C
C            CALCULATING REDUCED COSTS
C
        DO 987 I=1,M
        ITRIPC(I)=0
 987    CONTINUE
        IPNTR=1
        DO 960 I=1,M
        IZEN=ICPTR(I)
        DO 961 J=IPNTR,IZEN
        ITRIPC(I)=ITRIPC(I)+IFIX(FTCOS(ICTRP(J))+.5)
 961    CONTINUE
        IPNTR=ICPTR(I)+1
 960    CONTINUE
C
C             CREATING CLUSTER ARCS
C
        DO 929 I=1,M
        G(5*M+L,1)=5*M+L
        G(5*M+L,2)=I
        G(5*M+L,3)=I+M
        G(5*M+L,4)=0
        G(5*M+L,5)=ICN(I)
        G(5*M+L,6)=S(I,I)-ITRIPC(I)
        L=L+1
 929    CONTINUE
```

```
C
C          CREATE RETURN ARC
C
      G([illegible]L,1)=[illegible]
      G([illegible]L,2)=[illegible]2
      G([illegible]L,3)=[illegible]
      G([illegible]L,4)=[illegible]
      G([illegible]L,5)=[illegible]
      G([illegible]L,6)=[illegible]
      [illegible]2
      N9=3*[illegible]+L+2[illegible]
      CALL CCKNIG([illegible],N9,ICPT)
C     TEST FEASIBILITY
      IF(ICPT.EQ.9999999)GO TO 9999
C     SAVE FLOW IN RETURN ARC
      K20=F(N9)
      L=1
      DO 60000 I=1,K20
      NPATH(I)=L
C     START WITH SUPER SOURCE
      N1=N5
C     LOCATE THE NEXT NODE IN THE PATH
60001 DO 61000 J=1,N9
      IF(NS(J).NE.N1)GO TO 61000
C     DOES THE ARC HAVE FLOW?
      IF(F(J).GT.0)GO TO 62000
61000 CONTINUE
C     ERROR IF WE  PASS THROUGH THE LOOP
      GO TO 9999
62000 N2=NT(J)
      F(J)=F(J)-1
      IF(J.LT.N7)GO TO 63000
      PATH(L)=J
      L=L+1
63000 N1=N2
      IF(N1.NE.N8)GO TO 60001
      F(N9)=F(N9)-1
60000 CONTINUE
      NPATH(K20+1)=L
      F(N9)=K20
C
C               GENERATING NEW COLUMN FOR SET COVER
C
      INCRY=INCCL
      K1=F(N9)
      DO 970 I=1,K1
      INCRY=INCRY+1
      K2=NPATH(I)
      K3=NPATH(I+1)-1
      DO 971 J=K2,K3
      IRCL=PATH(J)-N7+1
      ITPTR=0
      IF((IRCL-1).GT.0) ITPTR=ICPTR(IRCL-1)
      IK6=ITPTR+1
      IK7=ICPTR(IRCL)
      DO 975 IK=IK6,IK7
      ISCCL(INCRY,ICTRP(IK))=1
975   CONTINUE
971   CONTINUE
970   CONTINUE
      DO 1075 IP=1,[illegible]
1075  [illegible]PATH(IP)=PATH(IP)-N7+1
      DO [illegible] I=1,K1
      INCCL=INCCL+1
```

```
      J=[illegible]
      [illegible]=S([illegible])+S(PATH(J),[illegible])
1[illegible]  IF([illegible]) GO TO [illegible]
      J=J+1
      [illegible](PATH(J-1),PATH(J))+S([illegible](J),PATH([illegible]))
      GO TO [illegible]
07[illegible]  ISCO[illegible]+S(PATH(J),P+1)
C
C     [illegible]
C
      PRINT [illegible]
      DO [illegible] I=1,[illegible]
      K2=[illegible](I)
      K3=[illegible](I+1)-1
      PRINT 1000, K3-K2+1
      DO 1010 J=K2,K3
      PRINT 1000, PATH(J)
1000  FORMAT (I4,[illegible])
1010  CONTINUE
9999  RETURN
      END
      SUBROUTINE COVER
      COMMON ITXOR(50),ITYOR(50),ITXOT(50),ITYOT(50),ITLOD(50),
     *       RTCOS(50),ICXOR(50),ICYOR(50),ICXOT(50),ICYOT(50),
     *       ICPTR(50),ICTRP(250),ISCOL(100,50),ISCOS(100),
     *       INTRP,INCLS,INCOL
      DIMENSION RSCOS(50),ICLHS(50),ICLHS(50),ICSOL(50),IOSOL(50),
     *          ISCOV(50)
      DATA INITR/10/
C
C     FUNCTION 7 - SET COVER
C
C     ENTER OUTPUT PARAMETER
C
      READ*, ITSET
C
C     INITIALIZE SOLUTION & COSTS
C
      IF(ITSET.EQ.2) GO TO 090
      IOCAR=IOCOS=0
      DO 080 I=1,INCOL
      ITCOV=0
      DO 070 J=1,INTRP
      IF(ISCOL(I,J).EQ.0) GO TO 070
      ITCOV=ITCOV+1
  070 CONTINUE
      IF(ITCOV.GT.1) GO TO 080
      IOCAR=IOCAR+1
      RSCOS(IOCAR)=FLOAT(ISCOS(I))
      IOSOL(IOCAR)=I
      IOCOS=IOCOS+ISCOS(I)
      IOLHS(IOCAR)=1
      IF(IOCAR.EQ.INTRP) GO TO 105
  080 CONTINUE
  090 IOCOS=999999
      DO 100 I=1,INTRP
      RSCOS(I)=RTCOS(I)
  100 CONTINUE
      GO TO 105
C
C     GENERATE SET COVERS
C
  105 DO 350 INITR=1,INITR
      ICCAR=ITCOV=0
```

```
      DO 110 I=1,INTRP
      ICLHS(I)=0
  110 CONTINUE
C
C     COMPUTE COST FOR EACH COLUMN & SELECT
C     MINIMUM COST COLUMN
C
  115 RMCOS=1.0E10
      IMIND=0
      DO 130 I=1,INCOL
      IRCOV=0
      RRCOS=FLOAT(ISCOS(I))
      DO 120 J=1,INTRP
      IF(ISCOL(I,J).EQ.0) GO TO 120
      IF(ICLHS(J).GT.0) GO TO 120
      IRCOV=IRCOV+1
      RRCOS=RRCOS-PSCOS(J)
  120 CONTINUE
      IF(IRCOV.EQ.0) GO TO 130
      IF(RRCOS.GE.RMCOS) GO TO 130
      RMCOS=RRCOS
      IMIND=I
  130 CONTINUE
      IF(IMIND.EQ.0) GO TO 270
C
C     ADD MINIMUM COST COLUMN TO SOLUTION
C
      ICCAR=ICCAR+1
      ICSOL(ICCAR)=IMIND
C
C     UPDATE LEFT-HAND-SIDE
C
      DO 140 I=1,INTRP
      IF(ISCOL(IMIND,I).EQ.0) GO TO 140
      ICLHS(I)=ICLHS(I)+1
      IF(ICLHS(I).GT.1) GO TO 140
      ITCOV=ITCOV+1
  140 CONTINUE
      IF(ITCOV.LT.INTRP) GO TO 115
C
C     REDUCE SET COVER TO PRIME COVER
C
      DO 170 I=1,ICCAR
  145 ITCOV=0
      DO 150 J=1,INTRP
      IF(ISCOL(ICSOL(I),J).EQ.0) GO TO 150
      IF(ICLHS(J).GT.1) GO TO 150
      ITCOV=ITCOV+1
  150 CONTINUE
      IF(ITCOV.GT.0) GO TO 170
      DO 160 J=1,INTRP
      IF(ISCOL(ICSOL(I),J).EQ.0) GO TO 160
      ICLHS(J)=ICLHS(J)-1
  160 CONTINUE
      ICSOL(I)=ICSOL(ICCAR)
      ICCAR=ICCAR-1
      IF(I.LE.ICCAR) GO TO 145
  170 CONTINUE
C
C     COMPUTE COSTS FOR COVER
C
      ICCOS=0
      DO 180 I=1,ICCAR
      ICCOS=ICCOS+ISCOS(ICSOL(I))
```

```
      IT COV=0
      DO 190 J=1,[illegible]
      IF(ISCOL(IOSOL(I),J).[illegible]) GO TO 190
      IF(ICLPS(J).GT.[illegible]) GO TO [illegible]
      ITCOV=ITCOV+1
  190 CONTINUE
      DO 200 J=1,INTRP
      IF(ISCOL(IOSOL(I),J).[illegible]) GO TO [illegible]
      IF(ICLPS(J).[illegible]) GO TO [illegible]
      RTCOS(J)=[illegible]
      GO TO 200
  190 RTCOS(J)=FLOAT(ISCOS(IOSOL(I)))/FLOAT(ITCOV)
  200 CONTINUE
  210 CONTINUE
C
C     STORE LEAST COST COVER
C
      IF(ICOOS.GE.ICOOS) GO TO 250
      DO 230 I=1,ICCAR
      IOSOL(I)=IOSOL(I)
  230 CONTINUE
      ICOOS=ICCOS
      ICCAR=ICCAR
      DO 240 I=1,INTRP
      ICLPS(I)=ICLPS(I)
  240 CONTINUE
  250 CONTINUE
C
C     REDUCE LEAST COST COVER TO PARTITION
C
      DO 270 I=1,INTRP
      IF(ICLPS(I).EQ.1) GO TO 270
      DO 260 J=1,ICCAR
      IF(ISCOL(IOSOL(J),I).EQ.0) GO TO 260
      ISCOL(IOSOL(J),I)=-1
      ICLPS(I)=ICLPS(I)-1
      IF(ICLPS(I).EQ.1) GO TO 270
  260 CONTINUE
  270 CONTINUE
C
C     COMPUTE COSTS FOR PARTITION
C
      DO 300 I=1,ICCAR
      ISCOV(I)=0
      DO 280 J=1,INTRP
      IF(ISCOL(IOSOL(I),J).NE.1) GO TO 280
      ISCOV(I)=ISCOV(I)+1
  280 CONTINUE
      DO 290 J=1,INTRP
      IF(ISCOL(IOSOL(I),J).NE.1) GO TO 290
      RTCOS(J)=FLOAT(ISCOS(IOSOL(I)))/FLOAT(ISCOV(I))
  290 CONTINUE
  300 CONTINUE
C
C     RETURN PARTITION
C
      PRINT 320, ICCAR
      DO 340 I=1,ICCAR
      IF([illegible].EQ.1)
     *      PRINT 320, ISCOV(I)
      IF([illegible].EQ.0)
     *      PRINT 320, IOSOL(I)
      DO 330 J=1,INTRP
      IF(ISCOL(IOSOL(I),J).EQ.0) GO TO 330
```

```
      IF(ISCOL([illegible](I),J).[illegible].0.1) GO TO 310
      IF(I.[illegible])
     *      [illegible]
      IF([illegible])
     *      ISCOL([illegible](I),J)=1
      GO TO [illegible]
  31. IF([illegible])
     *      [illegible]
  320 FORMAT [illegible]
  330 CONTINUE
  340 CONTINUE
      RETURN
C
C     RETURN UNCOVERED [illegible]OWS
C
  370 PRINT 320, [illegible]COV-INTRP
      DO 380 I=1,INTRP
      IF(ICLPS(I).GT.0) GO TO 380
      PRINT 320, I
  380 CONTINUE
      RETURN
      END
```

B-4

Chromatics (BASIC)
Data Generation Programs
and Menu Data Sets for
Delivery

```
[illegible] PRINT A-?
1000 REM *** INITIALIZATION
[illegible]
[illegible] GOSUB 11000
[illegible] 11151211" 3000,300 [illegible]"
1020 PRINT [illegible]
1100 REM *** INITIALIZE WORK AREAS
[illegible]
[illegible]
11[illegible] FOR I=1 TO 50
[illegible] FOR J=1 TO 3
1150 JN(I,J)=0:TP(I,J)=0
1160 NEXT J
1170 FOR J=4 TO 5
1180 JN(I,J)=0
1190 NEXT J
1200 NEXT I
1250 OUT&H90,0
1260 ON ERROR=2 GOTO 2000
1270 GOTO 1270
2000 REM *** LIGHT PEN INTERRUPT ROUTINE
2010 X=CURSX(4):Y=CURSY(4)
2020 IF Y>111 THEN 4000
2030 GOSUB 12000
2031 IF I=9 THEN 7000
2034 IF I=10 THEN 8000
2040 F1=I:CJ=0
2050 OUT&H90,0
2060 RESUME
4000 REM *** PLOT REGION LIGHT PEN HIT ROUTINE
4020 IF F1=0 OR F1>8 THEN 9900
4030 ON F1 GOTO 4200,4400,4600,4800,5000,5200,5400,5600
4040 OUT&H90,0
4050 RESUME
4200 REM *** DELETE JUNCTION
4210 GOSUB 15100
4220 IF I=0 THEN 9900
4221 IF JN(I,3)<>0 THEN E$="JUNCTION NOT ORDINARY":GOSUB 15000:GOTO 9900
4222 FOR J=1 TO MT:IF TP(J,1)=I OR TP(J,2)=I THEN 4224
4223 NEXT J:GOTO 4225
4224 E$="JUNCTION IN USE - NOT DELETED":GOSUB 15000:GOTO 9900
4225 C=0:DJ=I:GOSUB 15200:I=DJ
4230 FOR J=I TO MJ-1
4240 FOR K=1 TO 5
4250 JN(J,K)=JN(J+1,K)
4260 NEXT K
4270 NEXT J
4280 MJ=MJ-1
4290 FOR J=1 TO MT
4300 FOR K=1 TO 2
4310 IF TP(J,K)>I THEN TP(J,K)=TP(J,K)-1
4320 NEXT K
4330 NEXT J
4340 GOTO 9900
4400 REM *** ADD JUNCTION
4430 MJ=MJ+1
4440 JN(MJ,1)=X:JN(MJ,2)=Y:JN(MJ,3)=0:JN(MJ,4)=0:JN(MJ,5)=0
4450 DJ=MJ:C=1:GOSUB 15200
4460 PRINT CHR$(7);
4470 GOTO 9900
4600 REM *** DELETE FACILITY
4610 GOSUB 15100
4620 IF I=0 THEN 9900
4630 IF JN(I,3)=1 THEN 46[illegible]
```

```
4630 E$="JUNCTION NOT A FACILITY":GOSUB 15600
4640 GOTO 9900     'ERROR - THIS JUNCTION NOT A FACILITY
4650 C=5:DF=I:GOSUB 15300
4700 JN(DF,3)=0
4710 C=1:DJ=DF:GOSUB 15200
4720 GOTO 9900
4800 REM *** ADD FACILITY
4810 GOSUB 15100
4820 IF I=0 THEN 9900
4830 IF JN(I,3)=0 THEN 4850
4835 E$="JUNCTION NOT AVAILABLE FOR USE":GOSUB 15600
4840 GOTO 9900     'ERROR - THIS JUNCTION NOT AVAILABLE FOR NEW FACILITY
4850 C=2:DF=I:GOSUB 15300
4860 JN(DF,3)=1
4870 GOTO 9900
5000 REM *** DELETE DEMAND
5010 GOSUB 15100
5020 IF I=0 THEN 9900
5030 IF JN(I,3)=2 THEN 5080
5035 E$="JUNCTION NOT A DEMAND":GOSUB 15600
5040 GOTO 9900     'ERROR - THIS JUNCTION NOT DEMAND
5080 C=3:DC=I:GOSUB 15400
5090 JN(DC,3)=0
5100 C=1:DJ=DC:GOSUB 15200
5110 GOTO 9900
5200 REM *** ADD DEMAND
5210 GOSUB 15100
5220 IF I=0 THEN 9900
5230 IF JN(I,3)=0 THEN 5250
5235 E$="JUNCTION NOT AVAILABLE FOR USE":GOSUB 15600
5240 GOTO 9900     'ERROR - THIS JUNCTION NOT AVAILABLE FOR NEW DEMAND
5250 PRINT "^W1021,510110";CHR$(12);CHR$(21);
5260 INPUT "ORIGINAL DEMAND";JN(I,4)
5270 PRINT CHR$(12);"^G";
5280 C=4:DC=I:GOSUB 15400
5290 JN(DC,3)=2
5300 GOTO 9900
5400 REM *** DELETE ROUTE
5410 GOSUB 15100
5420 IF I=0 THEN 9900
5430 IF OJ=0 THEN OJ=I:GOTO 9900
5440 FOR J=1 TO MT
5450 IF TP(J,1)=OJ AND TP(J,2)=I THEN 5490
5460 IF TP(J,1)=I AND TP(J,2)=OJ THEN 5490
5470 NEXT J
5480 OJ=0:GOTO 9900
5490 C=8:DT=J:GOSUB 15500
5495 FOR K=J TO MT-1
5500 FOR L=1 TO 3
5510 TP(K,L)=TP(K+1,L)
5520 NEXT L
5530 NEXT K
5540 MT=MT-1
5550 OJ=0:GOTO 9900
5600 REM *** ADD ROUTE
5610 GOSUB 15100
5620 IF I=0 THEN 9900
5625 IF OJ<>0 THEN 5650
5630 OJ=I
5640 GOTO 9900
5650 MT=MT+1
5660 TP(MT,1)=OJ:TP(MT,2)=I
5670 PRINT "^W1021,510110";CHR$(12);CHR$(21);
5680 INPUT "ORIGINAL LENGTH OF ROUTE";TP(MT,3)
```

```
[illegible]
[illegible]
[illegible]
6710 GOTO [illegible]
7000 [illegible] SAVE [illegible]RY NETWORK
[illegible] PRINT "[illegible]";CHR$(21);CHR$(12)
[illegible] INPUT "[illegible] FILE NAME"[illegible]
[illegible]
[illegible]
[illegible] FOR I=1 TO [illegible]
[illegible] FOR J=1 TO 4
7070 PRINT #6: USING "[illegible]";JN(I,J);
7080 PRINT #6: ","
70[illegible] NEXT J
7090 PRINT #6: JN(I,5)
7100 NEXT I
7110 PRINT #6: USING "[illegible]";MT
7120 FOR I=1 TO MT
7130 FOR J=1 TO 2
7140 PRINT #6: USING "###";TP(I,J);
7150 PRINT #6: ",";
7160 NEXT J
7170 PRINT #6: TP(I,3)
7180 NEXT I
7190 DOS"CLOSE 6 "[illegible]
7192 GOSUB 20000
7195 PRINT CHR$(12);"~G~W1,1,510510"[illegible]CHR$(12);"[illegible]";
7200 F1=0:GOTO [illegible]
8000 REM *** RENEW DELIVERY NETWORK
8010 PRINT "~W3011,510110";CHR$(21);CHR$(12);
8020 INPUT "~C7INPUT FILE NAME"[illegible]R$
8030 DOS"OPEN 6 R "[illegible]".DAT"
8040 INPUT #6:[illegible]
8050 FOR I=1 TO MJ
8060 INPUT #6: JN(I,1),JN(I,2),JN(I,3),JN(I,4),JN(I,5)
8070 NEXT I
8080 INPUT #6:MT
8090 FOR I=1 TO MT
8100 INPUT #6: TP(I,1),TP(I,2),TP(I,3)
8110 NEXT I
8120 DOS"CLOSE 6"
8200 REM REDRAW NETWORK
8210 PRINT "~W1,112510510";CHR$(12);"[illegible]";
8220 FOR I=1 TO MT
8230 C=1:DT=I:GOSUB 15900
8240 NEXT I
8250 FOR I=1 TO MJ
8260 IF JN(I,3)=0 THEN C=1:DJ=I:GOSUB 15[illegible]
8270 IF JN(I,3)=1 THEN C=2:DF=I:GOSUB 1[illegible]
8280 IF JN(I,3)=2 THEN C=4:DD=I:GOSUB 15[illegible]
8290 NEXT I
8300 PRINT "~W3011,510110";CHR$(12);"[illegible]";CHR$(12);"~F~G";
8310 F1=0
8320 GOTO 9900
9900 REM *** LIGHT PEN INTERRUPT RETURN ROUTINE
9910 OUT[illegible],0
9920 RESUME
10000 REM *** COLOR MENU READ-IN ROUTINE
10010 DOS"OPEN 5 R "[illegible]
10020 INPUT #5: [illegible]
10030 DIM [illegible]
10040 FOR I=1 TO [illegible]
10050 INPUT #5: MENU(I,1),MENU(I,2),MENU(I,3),MENU(I,4),MENU(I,5)
10060 NEXT I
```

```
10[illegible] IF SMX = 0 THEN GOTO 1[illegible]
1[illegible] FOR I = 1 TO SMX
10[illegible] PRINT [illegible]MENU(I,1),MENU(I,[illegible]) [illegible]U(I,[illegible])(I)
10[illegible]0 NEXT I
10[illegible]0 [illegible]
10[illegible] RETURN
11000 REM *** MENU DRAW ROUTINE
1[illegible] PRINT "[illegible]";
1[illegible] FOR I=1 TO [illegible]
1[illegible] PRINT "[illegible]";
11040 PRINT USING "[illegible]";MENU(I,5);
11050 PRINT "[illegible]";
11060 PLOT MENU(I,1),MENU(I,2),MENU(I,3),MENU(I,4)
11070 PRINT "[illegible]";
11080 PLOT MENU(I,1),MENU(I,2),MENU(I,3),MENU(I,4)
11090 NEXT I
11100 PRINT CHR$(2[illegible]);
11110 IF SMX=0 THEN RETURN
11120 FOR I = 1 TO SMX
11130 PRINT "[illegible]";
11140 PLOT HDMENU(I,1),HDMENU(I,2)
11150 PRINT "[illegible]";
11160 PRINT USING "#"; HDMENU(I,3);
11170 PRINT CHR$(21);M$(I);
11180 NEXT I
11190 RETURN
12000 REM *** MENU LOCATE ROUTINE
12010 PRINT CHR$(27);"OA1";"[illegible]0,511[illegible]";CHR$(12);"";F";CHR$(27);"OA0";
12020 FOR I=1 TO MY
12030 IF X < MENU(I,1) OR X > MENU(I,3) THEN GOTO 12060
12040 IF Y < MENU(I,2) OR Y > MENU(I,4) THEN GOTO 12060
12050 GOTO 12090
12060 NEXT I
12070 I=0
12080 RETURN
12090 PRINT CHR$(7);
12100 PRINT "[illegible]C7[illegible]1";
12110 PLOT MENU(I,1),MENU(I,2),MENU(I,3),MENU(I,4)
12120 PRINT "[illegible]L[illegible]2";
12130 RETURN
15000 REM *** (15000) MACROS
15010 REM       15100 - LOCATE JUNCTION
15020 REM       15200 - DRAW JUNCTION DJ
15030 REM       15300 - DRAW FACILITY DF
15040 REM       15400 - DRAW DEMAND   DD
15050 REM       15500 - DRAW TRIP     DT
15060 REM       15600 - PRINT ERROR MESSAGE
15100 REM *** 15100 LOCATE JUNCTION
15110 IF MJ=0 THEN RETURN
15120 FOR I=1 TO MJ
15130 IF ABS(X-JN(I,1))<9 AND ABS(Y-JN(I,2))<9 THEN 15160
15140 NEXT I
15150 I=0
15155 RETURN
15160 PRINT CHR$(7);
15170 RETURN
15200 REM *** 15200 DRAW JUNCTION DJ, COLOR C
15210 PRINT "[illegible]";:PRINT USING "[illegible]";C;
15220 PRINT "[illegible]";:PLOT JN(DJ,1),JN(DJ,2):PRINT "[illegible]";
15230 RETURN
15300 REM *** 15300 DRAW FACILITY DF, COLOR C
15310 PRINT "[illegible]";:PRINT USING "[illegible]";C;
15320 PRINT "[illegible]";
15330 PLOT JN(DF,1)-8,JN(DF,2)-8,JN(DF,1)[illegible],JN(DF,2)[illegible]
```

```
15340 PRINT "~I";
15350 RETURN
15400 REM *** 15400 DRAW UP NODE   DO, COLOR C
15410 PRINT "~C";:PRINT USING "##";C;
15420 PRINT "~F";
15430 PLOT JN(DO,1),JN(DO,2):PRINT "~E~L";CHR$(21);"~U";
15440 PLOT JN(DO,1)-3,JN(DO,2)-10
15450 PRINT USING "##";N(DO,3);:PRINT "~G";
15460 RETURN
15500 REM *** 15500 DRAW TRIP      DT, COLOR C
15510 PRINT "~C";:PRINT USING "##";C;
15520 PRINT "~F";
15530 P1=TP(DT,1):P2=TP(DT,2)
15540 PLOT JN(P1,1),JN(P1,2),JN(P2,1),JN(P2,2)
15550 XM=(JN(P1,1)+JN(P2,1))/2
15560 YM=(JN(P1,2)+JN(P2,2))/2
15570 PRINT "~L";CHR$(21);"~U";:PLOT XM,YM:PRINT USING "##";TP(DT,3)
15580 PRINT "~G";
15590 RETURN
15600 REM *** 15600 DISPLAY ERROR E$ IN WINDOW
15610 PRINT "~N3011,510110~C7";CHR$(21);CHR$(12);E$;"~?320"
15620 PRINT CHR$(12);"~G";
15630 RETURN
20000 REM ***** SHORTEST PATH ALGORITHM *****
20100 REM INITIALIZATION OF THE DISTANCE MATRIX
20110 DIM D(MJ,MJ)
20120 FOR I=1 TO MJ:FOR J=I TO MJ:D(I,J)=99:D(J,I)=99:NEXT J:D(I,I)=0:NEXT I
20160 FOR K=1 TO MT:I=TP(K,1):J=TP(K,2)
20170 D(I,J)=TP(K,3):D(J,I)=TP(K,3)
20180 NEXT K
20190 PRINT "INITIALIZATION COMPLETE"
20195 PRINT MJ;" PASSES TO GO"
20200 REM *** SOLVE SHORTEST PATH ***
20210 FOR I=1 TO MJ:FOR J=1 TO MJ
20220 D(I,J)=100*D(I,J)+I
20230 NEXT J:NEXT I
20240 FOR I=1 TO MJ:FOR J=1 TO MJ
20250 IF J=I THEN 20320
20260 IF D(J,I)>=9900 THEN 20320
20270 FOR K=1 TO MJ
20280 IF D(I,K)>=9900 THEN 20310
20290 S=(D(J,I)+D(I,K))/100:T=D(J,K)/100
20300 IF S<T THEN D(J,K)=S*100+(D(I,K)-INT(D(I,K)/100)*100)
20310 NEXT K
20320 NEXT J
20325 PRINT "PASS ";I;"COMPLETE"
20330 NEXT I
20340 DOS"ARYSAVE "+S$+" D"
20360 ERASE D
20370 RETURN
20400 REM *** PRINT PATH SOLUTION ***
20410 PRINT "~N0,0,510510~U0,500";CHR$(21);CHR$(12);
20430 FOR L1=1 TO MJ:FOR L2=1 TO MJ
20440 PRINT USING "##### ";D(L1,L2);
20445 NEXT L2:PRINT " ":NEXT L1
20450 INPUT "~C7CONTINUE";A$
20460 RETURN
```

```
13,16
5,15,55,40,4
5,55,55,80,2
65,15,115,40,4
65,55,115,80,2
125,15,175,40,4
125,55,175,80,2
185,15,235,40,4
185,55,235,80,2
245,15,295,40,6
245,55,295,80,3
17,13,4,"DEL"
77,13,4,"DEL"
137,13,4,"DEL"
197,13,4,"DEL"
257,13,6,"SAVE"
17,53,2,"ADD"
77,53,2,"ADD"
137,53,2,"ADD"
197,53,2,"ADD"
254,53,3,"RENEW"
65,108,7,"DELIVERY PROBLEM GENERATOR"
6,95,1,"JUNCTION"
68,95,2,"FACILITY"
134,95,4,"DEMAND"
247,95,5,"ARCHIVE"
196,95,1,"ROUTE"
```

B-5

Chromatics (BASIC)
Display, Optimization and Data Manipulation
Programs and Menu Data Sets for
Delivery

```
[illegible] IF [illegible] THEN 3404
190 [illegible] 700
[illegible]
[illegible] DEF FNE(Z4,Z[illegible])=[illegible](Z4,1)-C(Z[illegible],1))^2+(C(Z[illegible],2)-C(Z[illegible],2))^2)
1000 '
1002 '     *****   INITIALIZATION   *****
1004 '
1005 PRINT CHR$(12)[illegible]
1010 GOSUB 10[illegible]
1020 PRINT "[illegible]"
1030 PRINT CHR$(12)[illegible] ROUTING"
1040 PRINT "[illegible]Gerald Tuma [illegible]"
1060 PRINT "[illegible]Frank R [illegible]"
1070 PRINT "[illegible]John J Jarvis"
1080 PRINT "[illegible]Don Ratliff"
1085 PRINT "[illegible]";
1090 GOSUB 800[illegible]
1100 M=1:GOSUB 11000
1110 OUT[illegible],0
2000 ' *** MENU 1 - INITIALIZATION
2001 '
2002 '  (2000) MENU 1 LIGHT PEN INTERRUPT ROUTINE
2003 '
2010 ON ERROR 2 GOTO 2030
2020 GOTO 2020
2030 X=CURSX(4):Y=CURSY(4)
2040 IF Y>111 THEN 9900
2050 GOSUB 12000
2060 IF I=1 THEN 2200
2070 IF I=2 THEN GOSUB 2500:GOTO 9900
2080 IF I=3 THEN 2700
2085 IF I=4 THEN 5000
2090 GOTO 9900
2200 '
2202 '     MENU 1  -  READ PARTIAL SOLUTION
2204 '
2210 PRINT "[illegible]2011,510110"07":CHR$([illegible]):CHR$(12);
2220 PRINT "INPUT PREVIOUS SOLUTION NA[illegible]"
2225 INPUT S$
2230 DOS"OPEN 5 R "+S$+" 75"
2240 INPUT#5:P$
2250 IF P$=R$ THEN 2282
2260 E$=S$+" IS A SOLN TO "+P$+", NOT "+R$:GOSUB 15000
2270 PRINT "[illegible]0,0,511511":CHR$(12)[illegible]
2275 DOS"CLOSE 5"
2280 GOTO 9900
2282 PRINT "[illegible]W1,1125105l0":CHR$(12);
2284 FOR I=1 TO MJ
2286 INPUT #5: JN(I,3),JN(I,4),JN(I,5),JN(I,3)
2294 NEXT I
2330 INPUT #5:AM,AL
2335 IF AM=0 THEN 2390
2340 FOR I=1 TO AM:INPUT #5: AR(I):NEXT I
2350 FOR I=1 TO AL:INPUT #5: AR(I):NEXT I
2390 DOS"CLOSE 5"
2392 X1=0:X2=0:Y1=0:Y2=0:GOSUB 5600:RS=0
2395 GOTO 2900
2500 '
2502 '     MENU 1  -  SAVE SOLUTION
2504 '
2510 PRINT "[illegible]2011,510110"07":CHR$([illegible]):CHR$(12);
2520 INPUT "SOLUTION OUTPUT NAME";S$
2530 DOS"OPEN 5 W "+S$+" 75"
2540 PRINT#5: R$
```

```
[illegible] PRINT PSJ(I,3)";"JN(I,[illegible]);"JN(I,5)";"JN(I,[illegible]) [illegible]
2580 [illegible]
2590 [illegible]
2600 FOR J=1 TO [illegible]:PRINT PS[illegible](I):NEXT I
2610 [illegible] TO [illegible]:PRINT PS[illegible](I):NEXT I
2640 [illegible]
2660 [illegible]
2670 PRINT [illegible];0,510510"[illegible](12)[illegible]
2675 [illegible]
2700 *
2702 *      MENU 1  -  PREPARE FOR A NEW SOLUTION
2704 *
2710 PRINT [illegible]12,510510"[illegible](12)[illegible]"G"07";
2720 FOR I=1 TO NJ
2725 IF JN(I,3)<>2 THEN 2750
2730 JN(I,[illegible]):DD=I:GOSUB 15400
2735 JN(I,6)=99:IF DF<>0 THEN JN(I,4)=2[illegible](I,DF)
2740 [illegible](I)=0
2750 NEXT I
2760 [illegible]=0:[illegible]:GT!=0:TS!=0:TC!=0
2900 *
2902 *      MENU 1  -  EXIT - ENTER MENU 2
2904 *
2910 PRINT "[illegible]1,1,510110";CHR$(12);"G"07"3300,[illegible]";
2920 M=2:GOSUB 11000
2930 [illegible]0,0
2940 F1=0
2950 RESUME 3000
3000 *
3002 *     (3000) MENU 2 - ROUTE-BUILDING MENU
3004 *
3006 *      MENU 2  -  LIGHT PEN INTERRUPT ROUTINE
3008 *
3010 ON ERROR=2 GOTO 3030
3020 GOTO 3020
3030 X=CURSX(4):Y=CURSY(4)
3040 IF Y>111 THEN 3100
3050 GOSUB 12000
3055 F2=F1:F1=I
3060 ON F1 GOTO 3300,3600,3500,4000,6000,3900
3070 GOTO 9900
3100 *
3102 *      MENU 2  -  PLOT REGION HIT ROUTINE
3104 *
3110 IF F1<>1 THEN 9900
3120 GOSUB 15100:IF I=0 THEN 9900
3130 IF MR<>0 THEN 3160
3135 IF JN(I,3)=1 THEN 3145
3140 E$="FIRST MUST BE A FACILITY":GOSUB 15600:GOTO 9900
3145 MR=MR+1:RT(MR)=I:DF=I:PRINT ""1";:C=2:GOSUB 15300:PRINT ""2";
3150 GOSUB 7000
3155 GOTO 9900
3160 IF JN(I,3)=0 THEN E$="BLANK JUNCTIONS DON'T COUNT":GOSUB 15600:GOTO 9900
3165 IF JN(I,3)=2 THEN 3200
3170 IF JN(I,3)=1 AND I<>RT(1) THEN E$="CANNOT [illegible]":GOSUB 15600:GOTO 9900
3172 IF MR<2 THEN E$="TRIVIAL ROUTE IGNORED":GOSUB 15600:GOTO 9900
3175 E$="ROUTE COMPLETE":GOSUB 15600
3180 MR=MR+1:RT(MR)=I:J1=RT(MR-1):J2=RT(MR)
3182 PRINT "[illegible]";:C=4:GOSUB 15700:PRINT "[illegible]";
3184 TC!=TC!+[illegible](J1,J2)
3186 GT!=GT!+TC!
3188 GOSUB 7200:GOTO 3600
3200 *
3202 *      MENU 2  -  BUILD - ALL [illegible] TO [illegible] ROUTE
```

```
3200 [illegible]
3205 J=1
3210 I [illegible] THEN [illegible] 15?00:GOTO 9400
3211 [illegible]
3212 [illegible] 15400
3213 [illegible] 15400
3215 [illegible] PLOT [illegible] 17
3220 PRINT [illegible] /100
3225 PRINT [illegible] 15700 [illegible]
3227 TC!=TC!+[illegible]
3230 TD!=TD!+[illegible] (5),32)
3235 TS!=TS!+JN(?,4)
3236 GOSUB 7000:GOSUB 7200
3240 GOTO 9900
3300 *
3302 *     MENU 2  -  BUILD - BEGIN ROUTE
3304 *
3310 GOSUB 15900
3730 GOTO 9900
3400 *
3402 *     MENU 2  -  BUILD - END ROUTE
3404 *
3410 IF RT(1)=RT(MR) THEN 3432
3420 [illegible]="ROUTE INCOMPLETE":GOSUB 15600:F1=1:GOTO 9?00
3430 PRINT CLR$(1):INPUT "OVERWRITE [illegible] (Y/N)";0$
3432 IF 0$="Y" THEN [illegible]:DOS"CHAIN BUILDOC"
3434 '****** WE REENTER HERE AFTER THE CHAINING ******
3435 GOSUB 15900
3440 FOR J=2 TO MR-1
3442 DD=RT(J):K=DD
3445 IF TD!=0 THEN TD!=1
3450 JN(K,5)=(TC!/TD!)*2*FND(DD,RT(1))
3455 C=3:IF DS=0 THEN C=8
3456 GOSUB 7000
3460 C=0:GOSUB 15400:C=6:DJ=DD:GOSUB 15200
3470 JN(K,5)=2
3480 NEXT J
3482 GOSUB 3700
3485 MR=0:TC!=0:TD!=0:TS!=0
3490 GOTO 3600
3500 *
3502 *     MENU 2  -  RESTART ROUTE
3504 *
3510 GOSUB 15900
3515 TC!=0:TD!=0:TS!=0:MR=0:AM=0:AL=0:ST=0
3520 FOR J=1 TO MJ
3525 IF JN(J,3)=2 THEN DD=J:C=4:GOSUB 15400:JN(J,5)=0
3530 NEXT J
3535 GOTO 3800
3600 *
3602 *     MENU 2  -  BACK-UP ROUTE
3604 *
3606 IF MR=0 THEN 9900 ELSE IF MR=1 THEN F1=1:GOTO 9900
3608 C=0:PRINT "[illegible]";:GOSUB 4700:PRINT "[illegible]";
3610 JN(RT(MR),5)=0
3612 DD=RT(MR):C=4:GOSUB 15400
3614 MR=MR-1
3616 C=7:PRINT "[illegible]";:GOSUB 4700:PRINT "[illegible]";
3618 F1=1:GOTO 9900
3700 *
3702 *     MENU 2  -  SAVE ROUTE IN ARCHIVE
3704 *
3710 AN=AN+1:A?(AN)=AL+1
3720 FOR J=1 TO MR-1
```

```
[illegible]
[illegible]
[illegible]
[illegible] '
[illegible] '     MENU 3  -  BUILD [illegible]
[illegible] '
[illegible] PRINT [illegible](12)[illegible]
[illegible] GOTO 9900
[illegible] '
[illegible] '     MENU 2  -  EXIT BACK TO MENU 1
[illegible] '
3960 GOSUB 1[illegible]00
[illegible] PRINT [illegible]110";CHR$(12)[illegible]
[illegible] M=1:GOSUB 11000
3980 OUT&H90,0
3985 RESUME 2000
4000 '
4010 '     MENU 3  -  INSERTION MENU SET-UP
4015 '
4030 PRINT "~H1,1,R10110";CHR$(12)[illegible]
4040 M=3:GOSUB 11000
4050 OUT&H90,0
4060 F1=0:J1=0:J2=0:J3=0:A1=0:A2=0:A3=0
4070 RESUME 4100
4100 '
4102 '     MENU 3  -  LIGHT PEN INTERRUPT ROUTINE
4104 '
4110 ON ERROR 2 GOTO 4130
4120 GOTO 4120
4130 X=CURSX(4):Y=CURSY(4)
4140 IF Y>111 THEN 4200
4150 PRINT "~H1,112D10510~;0";CHR$(12)[illegible]
4155 GOSUB 12000
4160 F1=I:IF F1=5 THEN GOTO 4600
4165 IF F1<>3 THEN J1=0:J2=0:J3=0:A1=0:A2=0:A3=0
4170 PRINT "~H1,111D10510~;0";CHR$(12)[illegible]
4180 IF F1=3 AND J1<>0 AND J2<>0 AND J3<>0 THEN 46[illegible]
4190 GOTO 9900
4200 '
4202 '     MENU 3  -  PLOT REGION HIT ROUTINE
4204 '
4210 IF F1=0 THEN 9900
4220 GOSUB 15100:IF I=0 THEN 9900
4230 IF M?>1 THEN 4250
4235 IF F1=3 OR F1=4 THEN 4250
4240 E$="ROUTE INSUFFICIENT LENGTH":GOSUB 15400:GOTO 9900
4250 ON F1 GOTO 4400,4300,4500,4750
4260 GOTO 9900
4300 '
4302 '     MENU 3  -  PRICE INSERT
4304 '
4310 GOSUB 4800:IF J=0 THEN 9900
4315 IF J1=0 THEN J1=J:GOTO 9900
4317 J2=J
4320 IF J1=J2+1 THEN J1=J2:J2=J2+1
4325 IF J1=J2-1 THEN Z=999:GOTO 4345
4330 E$="JUNCTIONS NOT ADJACENT":GOSUB 15400
4335 [illegible]:GOSUB 15200:IF J1<>0 THEN [illegible]:GOSUB 15200
4340 J1=0:J2=0:GOTO 9900
4345 FOR I=1 TO [illegible]
[illegible] IF JN(I,3)<>2 OR JN(I,1)<>0 THEN [illegible]
[illegible]
[illegible]
[illegible] IF [illegible] THEN Z=[illegible]
```

```
[illegible] 1
[illegible] GOSUB 15400 [illegible]
[illegible] 10520 "2";GOSUB [illegible] PRINT "~G~;F";
[illegible] GOTO 9900
4400 '
4402 '     MENU 3 -  LOCATE INSERT
4404 '
44[illegible] [illegible] THEN 9900
44[illegible] [illegible]
4420 FOR J=2 TO [illegible]
4425 [illegible]
4430 Z=[illegible]-FND(D1,D2)
44[illegible] IF Z<Z THEN Z=Z:J1=J:J2=J+1
4440 NEXT J
4442 D1=[illegible]
4443 Z=[illegible](D2,D2)-FND(D1,D2)
4444 IF Z<Z THEN Z=Z:J1=MR:J2=1
4445 IF Z=999 OR J1=0 OR J2=0 THEN 9900
4450 PRINT "~1~G";
4455 D=6:DJ=RT(J1):GOSUB 15200:DJ=RT(J2):GOSUB 15200:DD=J3:C=4:GOSUB 15400
4460 PRINT "~2~U3[illegible]51020"C/";CHR$(12,[illegible]);
4465 PRINT "COMPUTED SAVINGS ="Z
4470 PRINT FRE(0);" BYTES FREE"
4475 PRINT "~G";
4480 GOTO 9900
4500 '
4502 '     MENU 3 -  INSERT ROUTINE
4504 '
4505 IF MR<2 OR A1<>0 THEN 8600
4510 IF J1<>0 AND J2<>0 AND J3<>0 THEN 4600
4515 IF J1<>0 THEN 4530
4520 GOSUB 4[illegible]0:IF J=0 THEN 8600
4525 J1=J:DJ=RT(J):C=6:PRINT "~1";:GOSUB 15200:PRINT "~2";:GOTO 9900
4530 IF J3<>0 THEN 4545
4535 GOSUB 4870: IF J=0 THEN 9900
4540 J3=I:DD=I:C=4:PRINT "~1";:GOSUB 15[illegible]:PRINT "~2";:GOTO 9900
4545 GOSUB 4[illegible]00:IF J=0 THEN 9900
4547 IF (J1=J+1 OR J1=J-1) OR (J1=MR AND J=1) THEN 4555
4550 E$="INSERT POINTS NOT ADJ":GOSUB 1[illegible]:GOTO 9900
4555 J2=J:DJ=RT(J2):C=6:PRINT "~1";:GOSUB 15200:PRINT "~2";
4570 IF J2=J1-1 THEN J1=J1-1:J2=J1+1
4600 '
4602 '     MENU 3 -  PHYSICAL INSERTION OF J3 BETWEEN RT(J1) AND RT(J2)
4604 '
4610 C=0:PRINT "~4";
4615 T1=J1:T2=J2:T3=J3
4620 GOSUB 4700
4622 IF T2=1 THEN MR=MR+1:RT(MR)=T3:GOTO 4640
4625 FOR L=T2 TO MR
4630 RT(MR+T2-L+1)=RT(MR+T2-L)
4635 NEXT L
4637 MR=MR+1
4639 RT(T2)=T3
4640 PRINT "~;F";:DD=T3:C=0:GOSUB 15400:DJ=T3:C=6:GOSUB 15200
4642 PRINT "~C0";CHR$(21);"~U";:PLOT JN(T3,6)-5,JN(T3,7)+17:PRINT USING "####";
4644 JN(T3,5)=1
4645 PRINT "~;4";:C=0:GOSUB 4700
4650 J1=0:J2=0:J3=0
4655 PRINT "~U1,1125105100"B";CHR$(12);"~[illegible]";
4660 GOTO 9900
4700 '
4702 '     MENU 3 -  DRAW ROUTE IN RT() IN COLOR C
4704 '
4706 TC=0:T1=0:T0=0
```

```
4710 [illegible] (STEP [illegible]
47[illegible] [illegible] TO [illegible]
47[illegible] [illegible] TO [illegible]
47[illegible] [illegible] GOSUB [illegible]
47[illegible] [illegible]
47[illegible] [illegible]
47[illegible] [illegible](3),J)
47[illegible] [illegible]
4728 GOSUB 7100:GOSUB 7200
4730 RETURN
4750 '
4752 '      MENU 3  -  DELETE ITEM FROM ROUTE
4754 '
4756 IF M<>2 THEN [illegible]
4760 GOSUB 4800:IF J=0 THEN 4500
4765 IF J=1 THEN 5500
4770 [illegible]PRINT "[illegible]":GOSUB 47[illegible]:PRINT "[illegible]";
4775 J3=RT(J):D[illegible]:GOSUB 15[illegible]:IF J=NR THEN 4782
4780 FOR K=J+1 TO NR:RT(K-1)=RT(K):NEXT K
4782 NR=NR-1
4785 JN(J3,5)=0
4786 [illegible]PRINT "[illegible]":GOSUB 4700:PRINT "[illegible]";
4790 J1=0:J2=0:J3=0
4795 GOTO 5[illegible]
4800 '
4802 '      MENU 3  -  LOCATE JUNCTION ON CURRENT ROUTE
4804 '
4810 FOR J=1 TO NR
4820 IF RT(J)=I THEN 4850
4830 NEXT J
4840 E$="JUNCTION NOT IN CURRENT ROUTE":GOSUB 15000:J=0:RETURN
4850 PRINT CHR$(7);:D=0:DJ=I:PRINT "[illegible]1";:GOSUB 15[illegible]:PRINT "[illegible]2";
4860 RETURN
4870 '
4872 '      MENU 3  -  LOCATE DEMAND NOT ON CURRENT ROUTE
4874 '
4880 IF JN(I,3)=2 AND JN(I,5)=0 THEN J=I:RETURN
4882 J=0:E$="MUST BE UNSATISFIED DMD":GOSUB 15000:RETURN
4900 '
4902 '      MENU 3  -  EXIT BACK TO MENU 2
4904 '
4910 PRINT "[illegible]W1,1,510110";CHR$(12);"[illegible]G[illegible]C7[illegible]3300,330111";
4915 PRINT "[illegible]W1,1[illegible]10112";CHR$(12);"[illegible]F";
4920 M=2:GOSUB 11000
4930 OUT&H5D,0
4940 F1=1
4950 RESUME 3000
5000 '
5002 '      MENU 4  -  EDIT MENU SET-UP
5004 '
5010 PRINT "[illegible]W1,1,510110";CHR$(12);"[illegible]G[illegible]C7[illegible]3300,330111";
5015 M=4:GOSUB 11000
5020 OUT&H5D,0
5025 F1=0
5030 RESUME 5100
5100 '
5102 '      MENU 4  -  EDIT MENU LIGHT PEN INTERRUPT ROUTINE
5104 '
5110 ON ERROR 2 GOTO 5120
5115 GOTO 5110
5120 X=[illegible](4):Y=[illegible](4)
5125 IF Y>11[illegible] THEN 5200
5130 PRINT "[illegible]1,1[illegible]";CHR$(12);"[illegible]";
5175 GOSUB 120[illegible]
```

```
[illegible]
[illegible]
[illegible]
[illegible]
[illegible] 15000
[illegible]
[illegible]
[illegible] GOTO [illegible]
5200 *
5202 *     MENU 4  -  EDIT PLOT REGION [illegible]
5204 *
5206 IF [illegible] THEN [illegible]
5207 IF [illegible] THEN [illegible]
5208 GOSUB [illegible] THEN 9900
5210 ON [illegible] GOTO 5400,5400,5500
5212 GOTO 9900
5300 *
5302 *     MENU 4  -  EDIT DELETE
5304 *
5306 IF JN(I,3)=0 THEN 9900
5315 IF JN(I,3)=1 THEN 5370
5322 IF DS=1 THEN D=0:K=I:GOSUB 7400
5355 C=0:DF=I:GOSUB 15400:C=1:DJ=I:GOSUB 15200:JN(I,3)=0
5360 GOTO 9900
5370 C=0:DF=I:GOSUB 15300:C=1:DJ=I:GOSUB 15200:JN(I,3)=0
5375 GOTO 9900
5400 *
5402 *     MENU 4  -  EDIT ADD FACILITY
5404 *
5406 IF JN(I,3)<>0 THEN E$="JUNCTION NOT FREE":GOSUB 15600:GOTO 9900
5408 PRINT "~H3311,510110~C?"CHR$(21);CHR$(12);
5410 INPUT "~CENTER FACILITY CAPACITY";JN(I,4)
5412 PRINT CHR$(12);"~G";
5420 C=0:DJ=I:GOSUB 15200:C=2:DF=I:GOSUB 15400
5435 JN(I,3)=1
5440 GOTO 9900
5500 *
5502 *     MENU 4  -  EDIT ADD DEMAND
5504 *
5510 IF JN(I,3)<>0 THEN E$="JUNCTION NOT FREE":GOSUB 15600:GOTO 9900
5515 PRINT "~H3311,510110~C4";CHR$(21);CHR$(12);
5520 INPUT "~CENTER DEMAND AMOUNT";JN(I,4)
5525 PRINT CHR$(12);"~G";
5530 C=0:DJ=I:GOSUB 15200:C=4:DD=I:GOSUB 15400
5535 JN(I,3)=2
5545 JN(I,5)=0:JN(I,8)=99
5550 GOTO 9900
5600 *
5602 *     MENU 4 - SPECIFY ACTIVE ROUTE AREA
5604 *
5610 IF X1<>0 THEN 5620
5612 IF X2<>0 THEN 9900
5615 X1=X:Y1=Y:PRINT CHR$(7);"~G~C1X";:PLOT X1,Y1:GOTO 9900
5620 X2=X:Y2=Y
5625 X=X1 MAX X2:X1=X1 MIN X2:X2=X
5630 Y=Y1 MAX Y2:Y1=Y1 MIN Y2:Y2=Y
5635 PRINT "~$1~G#";:PLOT X1,Y1,X2,Y2:PRINT "~$F";
5640 CM=0:FOR I=1 TO NJ
5645 IF JN(I,6)<X1 OR JN(I,6)>X2 OR JN(I,7)<Y1 OR JN(I,7)>Y2 THEN 5655
5650 CM=CM+1:CA(CM)=I
5655 NEXT I
5699 GOTO 9900
5700 *
5702 *     MENU 4  -  REDRAW ACTIVE POINT [illegible]
```

```
[illegible]
[illegible]
[illegible]
[illegible]
[illegible] NEXT I [illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible] IF JR(J,3)=2 AND JR(J,3)=0 THEN [illegible] 19000
[illegible] THEN [illegible] 19400
5230 NEXT I
5240 RETURN
5900 *
5902 *     MENU 4  -  EXIT FORWARD TO MENU 2
5904 *
5906 GOTO 4900
6000 *
6002 *     MENU 5  -  DISPLAY MENU SET-UP
6004 *
6010 PRINT "[illegible]";CHR$(1[illegible]);"[illegible]3300,33011)";
6015 [illegible]GOSUB 11000
6020 [illegible]
6025 F1=0:J1=0:J2=0:J3=0:OP=0
6030 RETURN 6100
6100 *
6102 *     MENU 5  -  DISPLAY LIGHT PEN INTERRUPT ROUTINE
6104 *
6110 ON ERROR 2 GOTO 6120
6115 GOTO 6115
6120 X=CURSX(6):Y=CURSY(6)
6125 IF Y>111 THEN 6200
6130 PRINT "[illegible]";CHR$(12);"[illegible]"
6135 GOSUB 12000
6140 F1=1:IF F1=5 THEN 5900
6145 IF F1=1 THEN 6300
6147 IF F1=2 THEN 6400
6148 IF F1=3 THEN 6250
6150 GOTO 5900
6200 *
6202 *     MENU 5  -  DISPLAY PLOT REGION HIT ROUTINE
6204 *
6210 IF F1<>4 THEN 9900
6215 GOSUB 19100:IF I=0 THEN 9900
6200 GOTO 6500
6300 *
6302 *     MENU 5  -  DISPLAY PRICES (ON-OFF)
6304 *
6310 DS=1-DS:C=3:IF DS=0 THEN C=0
6315 FOR K2=1 TO CH:K=CA(K2):IF JR(K,3)=2 THEN GOSUB 7400
6317 NEXT K2
6320 GOTO 3300
6350 *
6355 *     MENU 5 - DISPLAY SAVINGS
6360 *
6365 DV=1-DV:C=2:IF DV=0 THEN C=0
6370 IF DV=1 THEN GOSUB [illegible]:GOSUB 9000
6375 PRINT "[illegible]";:GOSUB 15000:PRINT "[illegible]";
6380 GOTO 5900
6400 *
6402 *     MENU 5  -  DISPLAY JOINTS
6404 *
```

```
[illegible] ROUTE [illegible] 8:GOTO [illegible]
[illegible] GOSUB [illegible]
[illegible]
[illegible]
[illegible] C=1
[illegible]
[illegible]
[illegible]
[illegible] (K)=TT!
[illegible]
[illegible]
[illegible] GOSUB [illegible]
[illegible] THEN GOSUB 29700 ELSE PRINT [illegible] GOSUB 29740
6[illegible] GOTO [illegible]
6[illegible] *
6502 *     MENU 5  -  FREE ROUTE
6[illegible] *
6505 IF AM[illegible] THEN [illegible] ROUTES ARCHIV[illegible] GOSUB 1[illegible] GOTO [illegible]
650[illegible] GOSUB [illegible] IF [illegible] THEN 9900 ELSE C=[illegible]
6507 C=7:GOSUB 6700
6510 TT!=0:C=0:GOSUB 6700:GT!=GT!-TT!
6515 IF AP(K)=J3 THEN 9900
6520 FOR J=AP(K)+1 TO J3
6525 L=AR(J)
6530 J[illegible](L,[illegible])=0:C=[illegible]:GOSUB 15[illegible]
6535 NEXT J
6538 Z=AL-AP(AM)+1
6540 IF K=AM THEN 6555
6542 Z=AP(K+1)-AP(K)
6545 FOR L=AP(K+1) TO AL:AR(L-Z)=AR(L):NEXT L
6550 FOR L=K+1 TO AM:AP(L-1)=AP(L)-Z:NEXT L
6555 AL=AL-Z:AM=AM-1
6565 C=0
6590 GOTO 9900
6600 *
6602 *     MENU 5  -  DETERMINE ROUTE K OF JUNCTION I
6604 *
6610 FOR J=1 TO AL:IF AP(J)=I THEN 6625 ELSE NEXT J
6615 E$="JUNCTION NOT ARCHIVED":GOSUB 1[illegible]:K=0:RETURN
6625 FOR K=1 TO AM:IF AP(K)<= J THEN NEXT K
6630 K=K-1
6635 RETURN
6700 *
6702 *     MENU 5  -  DRAW ROUTE K, COLOR C
6704 *
6710 J3=AL:IF K<AM THEN J3=AP(K+1)-1
6712 TH!=TT!:TI!=TD!
6715 FOR J=AP(K) TO J3-1
6720 J1=AR(J):J2=AR(J+1):GOSUB 15700:TT!=TT!+FND(J1,J2)
6722 TD!=TD!+JN(J2,4)
6725 NEXT J
6730 J1=AR(J3):J2=AR(AP(K)):GOSUB 15700:TT!=TT!+FND(J1,J2)
6732 TH!=TT!-TH!:TI!=TD!-TI!
6735 RETURN
6900 *
6902 *     MENU 5  -  EXIT BACK TO MENU 2
6904 *
6905 IF [illegible]<>0 THEN K=0:PRINT "[illegible]":GOSUB 67[illegible]:PRINT "[illegible]":
6906 GOTO 4000
7000 *
7002 *    ***** AUXILIARY ROUTING ROUTINES
7004 *            7000 - [illegible]
7006 *            7200 - [illegible]
7008 *            7400 - [illegible] K, COLOR C
```

```
7008 '            7000 - POST OLD/NEW ROUTE [illegible]
7010 '
7015 [illegible] FROM [illegible]
7020 FOR [illegible] TO [illegible]
7025 [illegible] JN(I,4)<>0 THEN [illegible]
7030 [illegible]
7045 [illegible]
7050 NEXT [illegible]
7055 [illegible]
7060 GOSUB [illegible]
7065 PRINT [illegible]
7070 RETURN
7200 REM *** POST [illegible] IN WINDOW
7210 PRINT [illegible]
7218 PRINT"CLN[illegible] COST [illegible]
7222 PRINT [illegible]
7226 PRINT [illegible] TOTAL COST"[illegible]
7228 PRINT [illegible]
7230 PRINT ""G";
7240 RETURN
7400 REM *** POST [illegible] PRICE K, [illegible]
7405 PRINT ""G";:PRINT USING [illegible]
7410 PRINT [illegible]
7420 PLOT JN(K,6)*5,JN([illegible])*5
7430 PRINT USING [illegible];JN(K,8)/100;:PRINT ""G";
7440 RETURN
7600 '
7602 '     (7600)   POST OLD/NEW TOTALS IN WINDOW
7604 '
7610 PRINT ""M3211,5C0110"C7";CHR$(12);
7615 IF [illegible] THEN GOSUB 7200
7620 PRINT CHR$(21);
7625 PRINT "[illegible] OLD ROUTE COST "[illegible]
7630 PRINT "[illegible] OLD ROUTE [illegible]
7635 PRINT ""G";
7640 RETURN
8000 '
8002 '      *****  (8000)   AUXILIARY ROUTINES
8004 '
8006 '                 8000 - RENEW DELIVERY NETWORK
8007 '                 8200 - REDRAW NETWORK
8009 '
8010 PRINT ""M3211,510110";CHR$(21);CHR$([illegible]);
8020 INPUT ""J"C7INPUT PROBLEM FILE NAME"C4";R$
8030 DOS"OPEN 6 R "+R$+".DAT"
8040 INPUT #6:MJ,MF
8045 DIM JN(MJ,9),CP(MJ,2),CA(MJ)
8047 FOR I=1 TO MF:INPUT #6:CP(I,1),CP(I,2),JN(I,4):JN(I,3)=1:NEXT I
8049 INPUT #6:PD
8050 FOR I=MF+1 TO MJ
8060 INPUT #6: CP(I,1),CP(I,2),JN(I,4):JN(I,3)=2
8070 NEXT I
8080 DOS"CLOSE 6"
8100 '  SCALE DRAWING
8110 XS=9999:XL=0:YS=9999:YL=0
8115 FOR I=1 TO MJ
8120 XS=XS MIN CP(I,1)
8125 XL=XL MAX CP(I,1)
8130 YS=YS MIN CP(I,2)
8135 YL=YL MAX CP(I,2)
8140 NEXT I
8145 ST=(YL-YS) MAX ((XL-XS)/1.35)
8150 [illegible]
8155 FOR I=1 TO MJ
```

```
8[illegible] JN(I,1)[illegible]
8[illegible] JN(I,2)[illegible]YS)
8[illegible] JN(I,6)=JN([illegible]
8170 NEXT I
8[illegible] [illegible]J
8[illegible] [illegible] PROGRAM NETWORK
8[illegible] PRINT "[illegible]NJ,[illegible]2)[illegible]
8[illegible] [illegible]=0
8[illegible] FOR I=1 TO NJ
8260 IF JN(I,3)=0 THEN [illegible]
8270 IF JN(I,3)=1 THEN C=2[illegible]
82[illegible] IF JN(I,3)=2 THEN C=[illegible] 15[illegible]
8290 NEXT I
8300 PRINT "[illegible]K2[illegible],[illegible]CHR (12)[illegible]
8310 F1=0
8320 DIM CC([illegible]),RT([illegible]J+2),AR([illegible]J),AP([illegible])
8325 AM=0:AL=0
8330 J=0
8340 FOR I=1 TO NJ
8350 IF JN(I,3)=2 THEN JN(I,5)=0:JN(I,[illegible])
8360 NEXT I
8370 RETURN
8500 *
8502 *     (8500)  -  DELETE ITEM FROM ARCHIVED ROUTE
8504 *
8510 GOSUB 6[illegible]:IF K=0 THEN 9900
8515 C=0:GOSUB [illegible]:GOSUB [illegible]
8520 C=4:DD=I:GOSUB 15[illegible]:JN(I,5)=0
8525 *BEGIN ARCHIVE STACK PUSHDOWN
8530 IF K=AM THEN 8540
8535 FOR L=K+1 TO AM:AP(L)=AP(L)-1:NEXT L
8540 IF J=AL THEN 8550
8545 FOR L=J TO AL-1:AR(L)=AR(L+1):NEXT L
8550 AL=AL-1
8555 TT$=[illegible]:C=7:GOSUB [illegible]
8560 J1=0:J2=0:J3=0:A1=0:A2=0:A3=0
8565 GOSUB 7600
8570 GOTO 9900
8600 *
8602 *    (8600) - INSERT INTO AN ARCHIVED ROUTE
8604 *
8610 IF A1<>0 THEN 8625
8615 GOSUB 6600:IF K=0 THEN 9900 ELSE K1=K
8620 A1=J:DJ=AR(J):C=6:PRINT "~1";:GOSUB 15200:PRINT "~2";:GOTO 9900
8625 IF A3<>0 THEN 8640
8630 GOSUB 4870:IF J=0 THEN 9900
8635 A3=I:DD=I:C=4:PRINT "~1";:GOSUB 15400:PRINT "~2";:GOTO 9900
8640 GOSUB 6600:IF K=0 THEN 9900
8645 A2=J:DJ=AR(J):C=6:PRINT "~1";:GOSUB 15200:PRINT "~2";
8650 IF JN(AR(A1),3)<>1 OR JN(AR(A2),3)<>1 THEN [illegible]
8655 E$="BOTH ENDPOINTS FACILITIES":GOSUB 15600:GOTO 8750
8660 IF JN(AR(A2),3)=1 THEN K=K1:A2=AP(K1)
8665 IF JN(AR(A1),3)=1 THEN K1=K:A1=AP(K)
8670 IF K1<>K THEN E$="DIFFERENT ROUTES":GOSUB 15600:GOTO 8750
8675 IF A2<A1 THEN A4=A2:A2=A1:A1=A4
8677 IF A2=AL OR (K<>AM AND A2=AP(K+1)-1) THEN A1=A2-1
8678 IF ABS(A1-A2)=1 THEN 8[illegible]
8680 E$="INSERT POINTS NOT ADJ":GOSUB 15600:GOTO [illegible]
8685 IF A2<A1 THEN A1=A2:A2=A1+1
8690 C=0:GOSUB 6700:C=6:DJ [illegible]:GOSUB 15[illegible]:JN(AR[illegible]
8695 *BEGIN ARCHIVE EXPANSION
8700 IF K=AM THEN 8710
8705 FOR L=K+1 TO AM:AP(L)=AP(L)+1:NEXT L
8710 FOR L=A2 TO AL:AR(AL+1-(L-A2))=AR(AL-[illegible]):NEXT L
```

```
[illegible]
8770 [illegible] GOSUB 6100
87[illegible] PRINT [illegible]
87[illegible]
8760 GOSUB 7000
8790 GOTO 9000
9000 '
9002 '     ***** [illegible] LIGHT [illegible]
9004 '
9010 OPEN [illegible]
9020 RESUME
10000 '
10001 '     ***** COLOR MENU SERVICE ROUTINES
10002 '
10003 '              10000 - MENU READ-IN ROUTINE
10004 '              11000 - MENU DRAW ROUTINE (MENU M)
10005 '              12000 - MENU LOCATE ROUTINE
10006 '
10008 '     (10000) COLOR MENU READ-IN ROUTINE
10009 '
10010 DOS"OPEN 5 R [illegible]MENU4"
10020 INPUT #5; MN
10030 DIM M1(MN,4)
10040 M2=0:M3=0
10045 FOR I= 1 TO MN
10050 INPUT #5; X,Y
10060 M1(I,1)=M2+1:M1(I,2)=M2+X
10070 M1(I,3)=M3+1:M1(I,4)=M3+Y
10080 M2=M2+X:M3=M3+Y
10090 NEXT I
10100 DIM MB(M2,5),ML(M3,3),M[illegible](M3)
10110 FOR I=1 TO M2
10120 INPUT #5; MB(I,1),MB(I,2),MB(I,3),MB(I,4),MB(I,5)
10130 NEXT I
10140 FOR I=1 TO M3
10150 INPUT #5; ML(I,1),ML(I,2),ML(I,3),M[illegible](I)
10160 NEXT I
10170 DOS"CLOSE 5"
10180 RETURN
11000 '     (11000) COLOR MENU DRAW ROUTINE - MENU M ***
11010 PRINT "~C~K"
11020 FOR I=M1(M,1) TO M1(M,2)
11030 PRINT "~F~C";:PRINT USING "#";MB(I,5);
11040 PRINT "+";:PLOT MB(I,1),MB(I,2),MB(I,3),MB(I,4)
11050 PRINT "~C7~L+";:PLOT MB(I,1),MB(I,2),MB(I,3),MB(I,4)
11060 NEXT I
11070 PRINT CHR$(21);
11080 FOR I=M1(M,3) TO M1(M,4)
11090 PRINT "~U";:PLOT ML(I,1),ML(I,2):PRINT "~C";
11100 PRINT USING "#"; ML(I,3);:PRINT M[illegible](I);
11110 NEXT I
11120 PRINT "~G";
11130 RETURN
12000 '     (12000) COLOR MENU LOCATE ROUTINE - MENU M ***
12010 PRINT CHR$(27);"OA1~[illegible]";CHR$(12);"~[illegible]";CHR$(27);"OAG";
12020 FOR I=M1(M,1) TO M1(M,2)
12030 IF X < MB(I,1) OR X > MB(I,3) GOTO 12060
12040 IF Y < MB(I,2) OR Y > MB(I,4) GOTO 12060
12050 GOTO 12090
12060 NEXT I
12070 I=0:RETURN
12090 PRINT CHR$(7);"~C7~[illegible]";
12100 PLOT MB(I,1),MB(I,2),MB(I,3),MB(I,4)
12110 PRINT "~[illegible]L~2";
```

```
[illegible] [illegible](0,1)
[illegible] [illegible]
[illegible] *      (15000) MACROS
[illegible] *
[illegible] *              15100 - LOCATE JUNCTION
[illegible] *              15200 - DRAW JUNCTION [illegible]
[illegible] *              15300 - DRAW FACILITY [illegible]
[illegible] *              15400 - DRAW DEMAND [illegible]
[illegible] *              15600 - [illegible]
[illegible] *              15700 - DRAW SHORTEST [illegible]
[illegible] *              15800 - POST DEMAND [illegible]
[illegible] *              15900 - [illegible]
15100 *
15102 *      (15100) - MACRO - LOCATE JUNCTION I AT (X,Y)
15104 *
15110 IF CN=0 THEN RETURN
15120 FOR J=1 TO CN[illegible](J)
15130 IF ABS(X-JN(I,6))<5 AND ABS(Y-JN(I,7))<5 THEN 15160
15140 NEXT I
15150 I=0
15155 RETURN
15160 PRINT CHR$(7);
15170 RETURN
15200 *
15202 *      (15200) - MACRO - DRAW JUNCTION DJ, COLOR C
15204 *
15205 [illegible]
15210 PRINT "~C";:PRINT USING "#";C;
15220 PRINT "~F~";:PLOT JN(DJ,6),JN(DJ,7):PRINT "00S~L";
15230 RETURN
15300 *
15302 *      (15300) - MACRO - DRAW FACILITY DF, COLOR C
15304 *
15310 PRINT "~C";:PRINT USING "#";C;
15320 PRINT "~F";
15330 PLOT JN(DF,6)-5,JN(DF,7)-5,JN(DF,6)+[illegible],JN(DF,7)-5
15340 PRINT "~L";
15342 PRINT CHR$(21);"~L";:PLOT JN(DF,6)[illegible],JN(DF,7)-10
15344 PRINT USING "##";JN(DF,[illegible]);:PRINT "[illegible]";
15350 RETURN
15400 *
15402 *      (15400) - MACRO - DRAW DEMAND DD, COLOR C
15404 *
15410 PRINT "~C";:PRINT USING "#";C;
15420 PRINT "~F";
15430 PLOT JN(DD,6),JN(DD,7):PRINT "003~L";CHR$(21);"~U";
15440 PLOT JN(DD,6)-6,JN(DD,7)-10
15450 PRINT USING "###";JN(DD,4);:PRINT "~G";
15460 RETURN
15600 *
15602 *      (15600) - MACRO - DISPLAY [illegible] IN WINDOW
15604 *
15610 PRINT "~R3311,510110~C7";CHR$(21);CHR$(12);"[illegible]?02u"
15620 PRINT CHR$(12);"~G";
15630 RETURN
15700 *
15702 *      (15700) - DRAW SHORTEST PATH FROM J1 TO J2 IN COLOR C
15704 *
15710 PRINT "~C~C";:PRINT USING "#";C;:PRINT "[illegible]";
15740 PLOT JN(J1,6),JN(J1,7),JN(J2,6),JN(J2,7)
15770 RETURN
15800 *
15802 *      (15800) - POST SAVINGS  COLOR C
15804 *
```

```
[illegible] PRINT [illegible] [illegible]:
[illegible] [illegible]
[illegible] [illegible] THEN [illegible]
[illegible] [illegible]
[illegible] PRINT [illegible] INT [illegible]
15[illegible] [illegible]
15[illegible] [illegible]
15[illegible] [illegible]
15[illegible] RETURN
159[illegible] *
15[illegible] *     (159[illegible]) - [illegible] - [illegible] [illegible]
15[illegible] *
1590[illegible] PRINT "[illegible]510110"[illegible](12);
15[illegible] IF MR=0 THEN RETURN
159[illegible] PRINT "[illegible] 17[illegible]:PRINT "[illegible]";
15930 OF=[illegible](1):C=1:GOSUB 15[illegible]
15940 IF MR=1 THEN RETURN
15950 FOR J=2 TO [illegible]
15960 PRINT "[illegible]";J1=RT(J-1):J[illegible]=RT(J):[illegible]:PRINT "[illegible]";
15965 IF RT(J)<>RT(1) THEN C=F[illegible](J):C=[illegible]:GOSUB 15[illegible]
15970 JN(C,5)=0
15980 NEXT J
15990 RETURN
29700 *
29710 *   TEMP PROCEDURE - POST TOTAL INFO IN WINDOW
29720 *
29730 PRINT CHR$(21);"[illegible]510110";CHR$(12);"[illegible]";
29740 PRINT "[illegible]NUMBER OF POLICIES ";AM
29750 PRINT "[illegible]TOTAL COST ";GT;
29755 PRINT "[illegible]TOTAL DEMAND ";TD;
29756 C=0:PRINT:FOR K=1 TO AM
29758 C=C+1:IF C=7 THEN C=1
29760 PRINT "[illegible]";:PRINT USING "[illegible]";C;
29765 PRINT TP$(K);[illegible](K);"/";
29770 NEXT K
29775 PRINT "[illegible]G";
29780 RETURN
29800 *   TEMP PROCEDURE - POST ARCHIVE LIST IN WINDOW
29810 PRINT CHR$(21);"[illegible]W3311,510110";CHR$(12);"[illegible]C7";:PRINT:PRINT:PRINT
29820 PRINT AM;AL;"[illegible]C3"
29830 FOR GG=1 TO AM:PRINT AR(GG);:NEXT GG:PRINT "[illegible]C6"
29840 FOR GG=1 TO AL:PRINT AR(GG);:NEXT GG:PRINT
29845 PRINT "[illegible]G";
29850 RETURN
29900 *   TEMP PROCEDURE - POST ROUTE IN WINDOW
29910 PRINT CHR$(21);"[illegible]W3311,510110";CHR$(12);"[illegible]C7";:PRINT:PRINT:PRINT
29920 FOR GG=1 TO MR:PRINT RT(GG);:NEXT GG:PRINT
29930 PRINT T1;T2;[illegible]:PRINT "[illegible]G";
29940 RETURN
30000 * *** SIMPLE KEYBOARD DELAY W/O INTERRUPTS!
30005 PRINT CHR$(21);"[illegible]C[illegible] RETURN TO CONTINUE"
30010 PRINT CHR$(27);"OAF";:INPUT Z$:PRINT CHR$(27);"OAG";
30015 PRINT "[illegible]2[illegible]G";
30020 RETURN
31000 * *** TEST MENU ROUTINE
31002 CLEAR 1000:[illegible]="DLFMNU[illegible]":GOSUB 1[illegible]
31010 FOR [illegible]= 1 TO [illegible]
31020 PRINT "[illegible]0,0,511911[illegible]G";CHR$(12);
31030 GOSUB 11000
31040 PRINT CHR$(27);"OAF";:INPUT Z$:PRINT CHR$(27);"OAG";
31050 NEXT [illegible]
31060 STOP
```

```
[illegible]
[illegible] *     [illegible]
[illegible]
[illegible] *     [illegible] RESULC
[illegible] *     [illegible] A
[illegible] *     [illegible]
[illegible]       [illegible]
[illegible]
[illegible] *     [illegible]
[illegible]
[illegible] *     1. TRANSMIT A ROUTE TO THE CYBER [illegible] THE OPERATOR:
[illegible] *        A. NUMBER OF DELIVERY POINTS [illegible]
[illegible] *        B. X-Y COORDINATES OF EACH STOP
[illegible]
[illegible] *     2. THE CYBER SHOULD TRANSMIT BACK
[illegible] *        A. THE OPTIMAL ROUTE [illegible]
[illegible] *        B. THE SOLUTION FOR THE [illegible]
[illegible] *
[illegible] *     3. TO DETERMINE IF THE CYBER SOLUTION IS [illegible] TO
[illegible] *        THE OPERATOR-[illegible] SOLUTION. IF SO, WE REPLACE
[illegible] *        ROUTE ARCHIVE WITH THE CYBER SOLUTION
[illegible] *
2000 *
2010 *     (2000)  TRANSMIT THE ROUTE TO THE CYBER
[illegible] *
[illegible] PRINT CHR$(27)"RDA";CHR$(27)"[illegible]";CHR$(27)"RGA"; [illegible] UP COMMUNICATIONS
[illegible] PRINT USING "[illegible]";NR-1;
2050 PRINT #4[illegible]",";
2060 FOR J=1 TO NR-1
2070 PRINT #4;USING "###";CP(RT(J),1);
2080 PRINT #4;",";
2090 PRINT #4;USING "###";CP(RT(J),2);
2100 IF J<[illegible] THEN PRINT #4;",": ELSE PRINT #4;" "
2105 PRINT "C2";CP(RT(J),1);CP(RT(J),2);" ";
2110 NEXT J
2115 PRINT "C6ROUTE SENT TO CYBER"
2120 GOSUB 5000 *TRAP THE LINE FEED CHARACTER FROM THE CYBER
2130 PRINT "RETURN FROM PUNCH ROUTINE"
3000 *
3010 *     (3000)  RECEIVE ROUTE BACK FROM CYBER
3020 *
3030 DIM RR(NR-1,2)
3040 FOR J=1 TO NR-1
3047 GOSUB 7000
3050 INPUT#4;RR(J,1)
3055 RR(J,2)=RT(RR(J,1))
3057 PRINT CHR$(12);"C5";RR(J,1);"C3";RR(J,2);
3060 NEXT J
3065 GOSUB 7000
3070 INPUT#4;RC
3080 PRINT "C6CYBER ROUTE COST ";RC
3100 *
3110 *     (3100)  WE HAVE A NEW ROUTE TO REPLACE THE OLD ONE
3120 *
3125 PRINT "YOUR COST ";TC!
3127 IF RC=0 THEN PRINT "C4CYBER MESSED UP":GOTO 4000
3130 IF TC!>RC THEN GOTO 3170
3145 PRINT "C4YOU WON"
3150 GOTO 4000
3170 PRINT "C4CYBER WON"
3173 *REDRAW OLD ROUTE IN BLACK
3175 FOR J=1 TO NR-1
[illegible] PRINT "[illegible]";J1=RT(J):J2=RT(J+1):[illegible] ELSE PRINT "[illegible]";
3185 NEXT J
```

```
32?? *
3201 *     (3200)  PHYSICALLY REPLACE THE OLD ROUTE
3210 *
3220 FOR J=1 TO J0-1
3230 R(J)=P(J,2)
3250 NEXT J
3255 [illegible] THE NEW ROUTE IN RTC
3260 FOR J=1 TO J0-1
3265 [illegible]
3270 NEXT J
3280 [illegible]
3285 T=APC
3290 GOTO 6000
3800 *
3810 *     (3800)  DRAW SHORTEST PATH FROM J1 TO J2 IN COLOR C
3820 *
3830 PRINT "CO";:PRINT USING [illegible]
3840 PLOT JR(J1,6),JR(J1,7),JR(J2,6),JR(J2,7)
3850 RETURN
4000 *
4010 *     (4000)  RECHAIN BACK TO DLPFUC
4020 *
4025 ERASE RP
4030 DOS "CHAIN DLPFUC"
5000 *
5010 *     (5000)  TRAP LINE FEED CHARACTER FROM CYBER
5020 *
5025 PRINT CHR$(27);"IA4";
5030 K=INP(76)
5035 IF K<>10 THEN 5030
5050 OO=0  'PUNCH LF CHARACTER
5060 PRINT CHR$(27);"IA0";
5070 RETURN
6000 *
6010 *     (6000)  WAIT FOR ? IN INPUT STREAM
6020 *
6030 PRINT CHR$(27);"IA4";
6040 K=INP(76)
6050 IF K<>63 THEN 6040
6060 OO=0  'HOPEFULLY PUNCH ?
6070 PRINT CHR$(27);"IA0";
6080 RETURN
7000 *
7010 *     (7000)  PUNCH A SINGLE CHARACTER FROM THE SERIAL IO
7020 *
7030 PRINT CHR$(27);"IA4";
7040 OO=0
7050 PRINT CHR$(27);"IA0";
7060 RETURN
```

[illegible]

4,[illegible]

[illegible]

5,[illegible]

4,7

4,7

25,30,65,80,7

95,30,135,80,7

165,30,205,80,7

235,30,275,80,7

5,30,45,80,7

60,30,100,80,7

115,30,155,80,7

170,30,210,80,7

225,30,265,80,7

280,30,320,80,7

5,30,45,80,7

65,30,105,80,7

125,30,165,80,7

185,30,225,80,7

245,30,285,80,7

5,30,45,80,7

65,30,105,80,7

125,30,165,80,7

185,30,225,80,7

5,30,45,80,7

65,30,105,80,7

125,30,165,80,7

185,30,225,80,7

32,27,6,"[illegible]"

240,27,3,"[illegible]"

```
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible],95,7,"INTERACTIVE ROUTING - INITIALIZATION"
16,27,2,"[illegible]"
13,16,2,"ROUTE"
6[illegible],27,4,"[illegible]"
74,16,4,"UP"
115,27,5,"RESTART"
170,27,5,"INSERT"
225,27,7,"DISPLAY"
286,27,7,"EXIT"
6[illegible],95,7,"INTERACTIVE ROUTING - TOUR BUILDING"
7,27,6,"LOCATE"
7,16,6,"INSERT"
67,27,2,"PRICE"
63,16,2,"INSERT"
125,27,4,"INSERT"
186,27,7,"DELETE"
251,27,7,"EXIT"
2,95,7,"INTERACTIVE ROUTING - DEMAND INSERTION"
7,27,6,"DELETE"
75,27,2,"ADD"
60,16,2,"FACILITY"
135,27,4,"ADD"
125,16,4,"DEMAND"
[illegible],27,7,"EXIT"
2,95,7,"INTERACTIVE ROUTING - [illegible]"
7,27,2,"[illegible]"
```

B-6

Cyber (FORTRAN)
Optimization Programs for
Delivery

```
      PROGRAM ITS+(INPUT,OUTPUT,TAPE6=OUTPUT)
      DIMENSION IICOS(50,50),ICTOR(50),RTXOR(50),RTYOR(50)
      DATA IBIGM/99999999/
C
C     ENTER DATA
C
10    READ*, ITNUM
      DO 100 I=1,ITNUM
      READ*, RTXOR(I),RTYOR(I)
  100 CONTINUE
C
C     GENERATE COST MATRIX
C
      DO 110 I=1,ITNUM
      DO 110 J=1,ITNUM
      IICOS(I,J)=IFIX(100.0*SQRT((RTXOR(I)-RTXOR(J))**2+
     *                           (RTYOR(I)-RTYOR(J))**2))
      IF(I.EQ.J) IICOS(I,J)=IBIGM
  110 CONTINUE
C
C     SOLVE TSP
C
      CALL SOLVE1(ITNUM,IICOS,ICOCS,ICTOR)
C
C     PRINT SOLUTION
C
      DO 120 I=1,ITNUM
      IF(ICTOR(I).EQ.1) GO TO 130
  120 CONTINUE
130   CONTINUE
      DO 140 J=1,ITNUM
      WRITE(6,135) ICTOR(I)
135   FORMAT(I15)
      I=I+1
      IF(I.GT.ITNUM) I=1
  140 CONTINUE
      WRITE(6,135) ICOCS
      GO TO 10
      END
      SUBROUTINE SOLVE1(INNOD,IICOS,ICOOS,ICTOR)
C
C     THIS SUBROUTINE SOLVES THE PROBLEM USING
C     A LEAST-LOWER-BOUND BRANCHING RULE
C
      DIMENSION IICOS(50,50),ISTOR(2750),ICTOR(50),INMIS(50),
     *          ISPTR(275),IASST(50),ILAND(50),IIHAI(50),
     *          ICHAT(50),ILIST(50),IMARK(50),IZLIS(100),
     *          IREST(12)
      DATA IBIGM/99999999/,INSUB/275/,IMSTO/2750/,ICMAX/9999999/,
     *     IMASS/10000/
      RTIME=SECOND(CP)
      INASS=INSUB=IBPTR=0
      IIPTR=1
C
C     INITIALIZE SUBTOUR LIST
C
      ICPTR=1
      IINDU=IMSTO/(INNOD+5)-1
      DO 100 I=1,IINDU
      INPTR=ICPTR+INNOD+5
      ISTOR(ICPTR)=INPTR
      ICPTR=INPTR
  100 CONTINUE
      ISTOR(ICPTR)=0
```

```
      ICOCS=INNCOTI(MAX)+1
      IBEST(11)=0
      IBEST(10)=ICOCS
C
C     SOLVE ASSIGNMENT PROBLEM
C
  110 CALL HUNGARY(BB,100,INNCD,IICOS,IASST,ILABD,ILMAT,ICHAT,
     *                 ILIST,IMARK,IPLUS,IASS)
      IMCOS=IBEST(9)-IBEST(10)
      INASS=INASS+1
      IF(INASS.GE.IMASS) GO TO 400
  115 IF(IMCOS.GE.ICOCS) GO TO 270
C
C     FIND MINIMUM CARDINALITY SUBTOUR
C
      IMCAR1=IMCAR2=INNOD
      IMENT=1
      DO 120 I=1,INNOD
      INVIS(I)=0
  120 CONTINUE
  130 DO 140 I=1,INNOD
      IF(INVIS(I).EQ.0) GO TO 150
  140 CONTINUE
      GO TO 180
  150 INENT=ICENT=I
      ICCAR1=0
      DO 160 I=1,INNOD
      IPENT=INENT
      INENT=IASST(INENT)
      IF(INVIS(INENT).EQ.1) GO TO 170
      INVIS(INENT)=1
      IF(IICOS(IPENT,INENT).LT.0) ICCAR1=ICCAR1-1
  160 CONTINUE
  170 ICCAR1=ICCAR1+I-1
      ICCAR2=I-1
      IF(ICCAR1.GE.IMCAR1) GO TO 130
      IMCAR1=ICCAR1
      IMCAR2=ICCAR2
      IMENT=ICENT
      GO TO 130
  180 IF(IMCAR2.LT.INNOD) GO TO 240
C
C     SAVE MINIMUM COST TOUR
C
      ICOCS=IMCOS
      INENT=IMENT
      DO 190 I=1,INNOD
      INENT=IASST(INENT)
      IOTOR(I)=INENT
  190 CONTINUE
      IF(INSUB.EQ.0) GO TO 270
C
C     REMOVE FATHOMED SUBPROBLEMS
C
      DO 230 I=1,INSUB
  200 IF(ISTOR(ISPTR(I)+2).LT.ICOCS) GO TO 230
      IRPTR=ISPTR(I)
  210 IPPTR=ISTOR(IRPTR)
      ISTOR(IRPTR)=IIPTR
      IIPTR=IRPTR
      IF(IPPTR.EQ.0) GO TO 220
      ISTOR(IPPTR+3)=ISTOR(IPPTR+3)-1
      IF(ISTOR(IPPTR+3).GT.0.OR.IPPTR.EQ.IPRTF) GO TO 220
      IRPTR=IPPTR
```

```
      GO TO 210
  220 ISPTR(I)=ISPTR(INSUB)
      INSUB=INSUB-1
      IF(I.LE.INSUB) GO TO 200
  230 CONTINUE
      GO TO 270
C
C     ADD SUBTOUR TO LIST
C
  240 IF(IIPTR.GT.0) GO TO 250
      GO TO 400
  250 IAPTR=IIPTR
      IIPTR=ISTOR(IIPTR)
      IF(INSUB.LT.IMSUB) GO TO 260
      GO TO 400
  260 INSUB=INSUB+1
      ISPTR(INSUB)=IAPTR
      ISTOR(IAPTR)=IBPTR
      ISTOR(IAPTR+1)=ISNOD
      ISTOR(IAPTR+2)=IMCOS
      ISTOR(IAPTR+3)=0
      ISTOR(IAPTR+4)=IMCAR2
      INENT=INEXT
      DO 265 I=1,IMCAR2
      IPENT=INENT
      INENT=IASST(IPENT)
      IF(IICOS(IPENT,INENT).LT.0) IPENT=-IPENT
      ISTOR(IAPTR+I+4)=IPENT
  265 CONTINUE
      IF(IBPTR.EQ.0) GO TO 315
      ISTOR(IBPTR+3)=ISTOR(IBPTR+3)+1
C
C     RESTORE ARCS
C
  270 IF(IBPTR.EQ.0) GO TO 315
      IIROW=ISTOR(IBPTR+ISNOD+4)
      IICOL=IABS(ISTOR(IBPTR+ITNOD+4))
      IICOS(IIROW,IICOL)=IICOS(IIROW,IICOL)-IBIGM
      IF((ILAND(IIROW)+ICHAT(IICOL)).LE.IICOS(IIROW,IICOL))
     *      GO TO 275
      IREST(9)=IREST(9)+IICOS(IIROW,IICOL)
     *                   -ILAND(IIROW)-ICHAT(IICOL)
      ILAND(IIROW)=IICOS(IIROW,IICOL)-ICHAT(IICOL)
  275 IPNOD=ISNOD
      ISNOD=ISNOD+1
      IF(ISNOD.LE.ISCAR) GO TO 350
      IINOU=ISCAR-1
      DO 280 I=1,IINOU
      IIROW=ISTOR(IBPTR+I+4)
      IF(IIROW.LT.0) GO TO 280
      IICOL=IABS(ISTOR(IBPTR+I+5))
      IICOS(IIROW,IICOL)=IICOS(IIROW,IICOL)+IBIGM
      IREST(9)=IREST(9)-IBIGM
  280 CONTINUE
      ISNOD=ISTOR(IBPTR+1)
      IPPTR=ISTOR(IBPTR)
  285 IF(IPPTR.EQ.0) GO TO 305
      ISCAR=ISTOR(IPPTR+4)
      IIROW=ISTOR(IPPTR+ISNOD+4)
      ITNOD=ISNOD+1
      IF(ITNOD.GT.ISCAR) ITNOD=1
      IICOL=IABS(ISTOR(IPPTR+ITNOD+4))
      IICOS(IIROW,IICOL)=IICOS(IIROW,IICOL)-IBIGM
      IF((ILAND(IIROW)+ICHAT(IICOL)).LE.IICOS(IIROW,IICOL))
```

```
     *        GO TO 290
           IBEST(9)=IBEST(9)+IICOS(IIROW,IICOL)
     *                    -ILAMD(IIROW)-ICHAT(IICOL)
           ILAMD(IIROW)=IICOS(IIROW,IICOL)-ICHAT(IICOL)
   290 IINCU=ISNOD-1
           IF(IINCU.LT.1) GO TO 300
           DO 295 I=1,IINCU
           IIROW=ISTOR(IPPTR+I+4)
           IF(IIROW.LT.0) GO TO 295
           IICOL=IABS(ISTOR(IPPTR+I+5))
           IICOS(IIROW,IICOL)=IICOS(IIROW,IICOL)+IBIGM
           IBEST(9)=IBEST(9)-IBIGM
   295 CONTINUE
   300 ISNOD=ISTOR(IPPTR+1)
           IPPTR=ISTOR(IPPTR)
           GO TO 285
C
C          REMOVE UNBRANCHED PROBLEM
C
   305 IPPTR=IBPTR
   310 IF(ISTOR(IPPTR+3).GT.0) GO TO 315
           IBPTR=IPPTR
           IPPTR=ISTOR(IBPTR)
           ISTOR(IBPTR)=IIPTR
           IIPTR=IBPTR
           IF(IPPTR.EQ.0) GO TO 315
           ISTOR(IPPTR+3)=ISTOR(IPPTR+3)-1
           GO TO 310
C
C          BRANCH ON LEAST LOWER BOUND SUBPROBLEM
C
   315 IF(INSUB.EQ.0) GO TO 400
           IBPTR=0
           IBCOS=IBIGM
           DO 320 I=1,INSUB
           IF(ISTOR(ISPTR(I)+2).GE.IBCOS) GO TO 320
           IBPTR=ISPTR(I)
           IBIND=I
           IBCOS=ISTOR(IBPTR+2)
   320 CONTINUE
           IF(IBPTR.EQ.0) GO TO 400
           ISPTR(IBIND)=ISPTR(INSUB)
           INSUB=INSUB-1
C
C          PROHIBIT AND REQUIRE ARCS
C
           ISNOD=ISTOR(IBPTR+1)
           IPPTR=ISTOR(IBPTR)
   330 IF(IPPTR.EQ.0) GO TO 350
           ISCAR=ISTOR(IPPTR+4)
           IIROW=ISTOR(IPPTR+ISNOD+4)
           ITNOD=ISNOD+1
           IF(ITNOD.GT.ISCAR) ITNOD=1
           IICOL=IABS(ISTOR(IPPTR+ITNOD+4))
           IICOS(IIROW,IICOL)=IICOS(IIROW,IICOL)+IBIGM
           IINCU=ISNOD-1
           IF(IINCU.LT.1) GO TO 345
           DO 340 I=1,IINCU
           IIROW=ISTOR(IPPTR+I+4)
           IF(IIROW.LT.0) GO TO 340
           IICOL=IABS(ISTOR(IPPTR+I+5))
           IICOS(IIROW,IICOL)=IICOS(IIROW,IICOL)-IBIGM
           IBEST(9)=IBEST(9)+IBIGM
           IF((ILAMD(IIROW)+ICHAT(IICOL)).LE.IICOS(IIROW,IICOL))
```

```
     *      GO TO 350
      IREST(9)=IREST(9)+IICOS(IIROW,IICOL)
     *                 -ILAMB(IIROW)-ICHAT(IICOL)
      ILAMB(IIROW)=IICOS(IIROW,IICOL)-ICHAT(IICOL)
  340 CONTINUE
  345 ISNOD=ISTOR(IFPTR+2)
      IFPTR=ISTOR(IFPTR)
      GO TO 3[illegible]
  350 ISNOD=1
      IFNOD=0
      ISCAN=ISTOR(IFPTR+4)
  360 IIROW=ISTOR(IFPTR+ISNOD+4)
      IF(IIROW.LT.0) GO TO 365
      ISNOD=ISNOD+1
      GO TO 360
  365 IFNOD=ISNOD+1
      IF(IFNOD.GT.ISCAN) IFNOD=1
      IICOL=IABS(ISTOR(IFPTR+IFNOD+4))
      IICOS(IIROW,IICOL)=IICOS(IIROW,IICOL)+ISIGM
      IF(IFNOD.EQ.0) GO TO 375
      IIROW=ISTOR(IFPTR+IFNOD+4)
      IF(IIROW.LT.0) GO TO 370
      IICOL=IABS(ISTOR(IFPTR+IFNOD+5))
      IICOS(IIROW,IICOL)=IICOS(IIROW,IICOL)-ISIGM
      IREST(9)=IREST(9)+ISIGM
      IF((ILAMB(IIROW)+ICHAT(IICOL)).LT.IICOS(IIROW,IICOL))
     *      GO TO 370
      IREST(9)=IREST(9)+IICOS(IIROW,IICOL)
     *                 -ILAMB(IIROW)-ICHAT(IICOL)
      ILAMB(IIROW)=IICOS(IIROW,IICOL)-ICHAT(IICOL)
  370 CONTINUE
  375 IREST(11)=1
      IREST(12)=ICCOS
      IF(IREST(9).LT.ICCOS) GO TO 110
      ICCOS=IREST(9)
      GO TO 110
C
C     RETURN
C
  400 RETURN
      END
C
C***********************************************************************
C
      SUBROUTINE HUNGARY(NMAX,ZMAX,N,C,A,LAMDA,IHAT,CHAT,LIST,MARKER,
     *                   ZLIST,RESULTS)
C
C------THIS SUBROUTINE SOLVES THE CLASSICAL ASSIGNMENT PROBLEM VIA
C------KUHN'S HUNGARIAN ALGORITHM.  IT USES A LABELLING AND LIST
C------PROCESSING PROCEDURE TO MAKE THE ALGORITHM MORE EFFICIENT
C------THAN THE STRAIGHTFORWARD MATRIX IMPLEMENTATION.  ALSO, THE
C------DUAL PHASE IS CONTINUED UNTIL A FLOW CHANGE RESULTS WHICH
C------ADDS A NEW ASSIGNMENT.  THE FLOW CHANGE OR PRIMAL PHASE IS
C------ACCOMPLISHED USING LABELS AND LISTS DEVELOPED IN THE DUAL
C------PHASE AND IS A ONE PASS PROCEDURE.
C
C***********************************************************************
C
      INTEGER C(NMAX),A(NMAX,2*NMAX),LAMDA(NMAX),IHAT(NMAX),CHAT(NMAX)
      INTEGER LIST(NMAX),MARKER(NMAX),ZLIST(ZMAX),RESULTS(12)
      INTEGER CHIN,ADDROW
      LOGICAL FULL
      INTEGER VAL1,ASGNTS,VAL2,MARK,DCIR,PPT,OIT,VAL3,NMAX
      VAL1=0
```

```
      ASNMTS=0
      VAL1=RESULT(20)
      NCUNT=0
      OPTN=0
      PIT=0
      DIT=0
      VAL2=0
      MULT=.FALSE.
      DO 900 I=1,N
      A(I)=0
  900 IHAT(I)=0
      IF(RESULT(11).EQ.1) GO TO 2000
      DO 950 I=1,N
      LAMDA(I)=0
  950 CHAT(I)=0
      CALL SECOND(TIM1)
C
C------INITIALIZE MULTIPLIER VECTORS
C
C------ROW MULTIPLIERS AND ASSIGNMENTS
      DO 1100 I=1,N
      CMIN=999999999
      JMIN=0
      DO 1000 J=1,N
      IF(C(I,J).GE.CMIN)GO TO 1000
      CMIN=C(I,J)
      JMIN=J
 1000 CONTINUE
      VAL1=VAL1+CMIN
      LAMDA(I)=CMIN
      IF(IHAT(JMIN).NE.0)GO TO 1100
      A(I)=JMIN
      IHAT(JMIN)=I
      ASNMTS=ASNMTS+1
 1100 CONTINUE
C-----COLUMN MULTIPLIERS AND ASSIGNMENTS
      DO 1200 J=1,N
      IF(IHAT(J).NE.0)GO TO 1200
      CMIN=999999999
      IMIN=0
      DO 1150 I=1,N
      IF((C(I,J)-LAMDA(I)).GE.CMIN)GO TO 1150
      CMIN=C(I,J)-LAMDA(I)
      IMIN=I
 1150 CONTINUE
      VAL1=VAL1+CMIN
      CHAT(J)=CMIN
      IF(A(IMIN).NE.0)GO TO 1200
      ASNMTS=ASNMTS+1
      A(IMIN)=J
      IHAT(J)=IMIN
 1200 CONTINUE
      VAL2=VAL1
      TIM1=SECOND(TIM2)-TIM1
C
C------PRIMAL PHASE:  STEP 0- INITIALIZE LIST
C
 2000 NROW=0
      DO 2100 I=1,N
      MARKER(I)=0
      IF(A(I).NE.0)GO TO 2100
      NROW=NROW+1
      LIST(NROW)=I
 2100 CONTINUE
```

```
      IF(NROW.EQ.0)GO TO 9000
      KROW=NROW
C
C-----PRIMAL PHASE: STEP 1- [illegible] ROWS AND
C-----LABEL COLUMNS.  ([illegible] FLOW [illegible])
C
      IFO.=1
      JSTRT=0
 3010 PIT=PIT+1
 3020 INOW=LIST(KROW)
      DO 3100 J=1,N
      IF(MARKER(J).NE.0)GO TO 3100
      IF((C(INOW,J)-LAMDA(INOW)-CHAT(J)).GT.0)GO TO 3100
C-----FORWARD LABEL
      MARKER(J)=INOW
C-----CHECK FOR PRIMAL BREAKTHROUGH
      IF(IMAT(J).NE.0)GO TO 3050
      JSTRT=J
      PBRK=PBRK+1
      GO TO 4000
C-----REVERSE LABEL
 3050 NROW=NROW+1
      LIST(NROW)=IMAT(J)
 3100 CONTINUE
      IROW=IROW+1
      IF(IROW.LE.NROW)GO TO 3020
      VAL3=KROW
      IF((VAL2+VAL3).GE.RESULTS(12)) GO TO 4000
      VAL3=0
C-----LABELING COMPLETE
C
C-----DUAL PHASE: STEP 1- FIND MINIMUM [illegible]
C-----COST IN MARKED ROWS AND UNMARKED COLUMNS
C
      DIT=DIT+1
      CMIN=999999999
      DO 3200 I=1,NROW
      INOW=LIST(I)
      DO 3150 J=1,N
      IF(MARKER(J).NE.0)GO TO 3150
      ITEST=C(INOW,J)-LAMDA(INOW)-CHAT(J)
      IF(ITEST-CMIN)3120,3130,3150
 3120 NZERO=1
      ZLIST(1)=INOW
      ZLIST(2)=J
      CMIN=ITEST
      FULL=.FALSE.
      JSTRT=0
      GO TO 3145
 3130 IF(NZERO.LT.(ZMAX/2))GO TO 3140
      FULL=.TRUE.
      GO TO 3145
 3140 NZERO=NZERO+1
      ZLIST(2*NZERO-1)=INOW
      ZLIST(2*NZERO)=J
C-----NOTE DUAL BREAKTHROUGH IF IT OCCURS
 3145 IF(IMAT(J).NE.0)GO TO 3150
      JSTRT=J
      ISTRT=INOW
 3150 CONTINUE
 3200 CONTINUE
      VAL3=KROW-CMIN
      IF((VAL2+VAL3).GE.RESULTS(12)) GO TO 90[illegible]
      VAL3=0
```

```
C
C-----DUAL PHASE [illegible] UPDATE MULTIPLIERS AND
C-----ACCOUNT FOR [illegible] MARKERS
C
C-----COLUMN MULTIPLIERS
      DO 3300 J=1,N
      IF(MARKER(J)[illegible])GO TO 3300
      CHAT(J)=[illegible]
      VAL=VAL2[illegible]
 3300 CONTINUE
C-----ROW MULTIPLIERS
      ADDROW=0
      DO 3400 I=1,NROW
      LAMBDA(LIST(I))=LAMBDA(LIST(I))+OMIN
 3400 VAL=VAL+OMIN
C-----CHECK FOR DUAL BREAKTHROUGH
      IF(JFIRST.EQ.0)GO TO 3410
      DBRK=DBRK+1
      MARKER(JFIRST)=[illegible]
      GO TO 4000
 3410 CONTINUE
C-----UPDATE COLUMN MARKER LABELS
      DO 3500 K=1,NZERO
      JNOW=ZLIST(2*K)
      INOW=ZLIST(2*K-1)
      MARKER(JNOW)=INOW
      IF(ADDROW.EQ.0)GO TO 3450
      DO 3440 K1=1,ADDROW
 3440 IF(LIST(NROW+K1).EQ.IHAT(JNOW))GO TO 3500
 3450 ADDROW=ADDROW+1
      LIST(NROW+ADDROW)=IHAT(JNOW)
 3500 CONTINUE
C-----CHECK TO SEE IF ALL ZERO ARCS ARE IN ZLIST
      IF(.NOT.FULL)GO TO 3900
C-----START SCANNING C FROM LAST ENTRY IN ZLIST
      I1=ZLIST(ZMAX-1)
      J1=ZLIST(ZMAX)+1
      DO 3600 I=1,NROW
 3600 IF(I1.EQ.LIST(I))ITEMP=I
      I1=ITEMP
       IF(J1.NE.N)GO TO 3610
      I1=I1+1
      IF(I1.GT.NROW)GO TO 3900
      J1=1
 3610 DO 3800 K=I1,NROW
      I=LIST(K)
      DO 3700 J=J1,N
      IF(MARKER(J).NE.0)GO TO 3700
      IF((C(I,J)-LAMBDA(I)-CHAT(J)).GT.0)GO TO 3700
      MARKER(J)=I
      DO 3640 K1=1,ADDROW
 3640 IF(LIST(NROW+K1).EQ.IHAT(J))GO TO 3700
 3650 ADDROW=ADDROW+1
      LIST(ADDROW+NROW)=IHAT(J)
 3700 CONTINUE
      J1=1
 3800 CONTINUE
 3900 CONTINUE
      IROW=NROW+1
      NROW=NROW+ADDROW
C-----NORMAL [illegible] CONTINUE LABELLING
      GO TO 3000
C
C-----PRIMAL PHASE [illegible] ASSIGNMENT
```

```
C-----[illegible] JSTRT
C
 4000 JNXT=JSTRT
 4500 JNOW=JNXT
      INOW=P(2,[illegible])-JNOW
      JNXT=[illegible](INOW)
      THAT(JNXT)=T[illegible]
      A(INOW)=[illegible]
      IF(JNXT.[illegible]) GO TO 9000
      GO TO 4500
C
C------TERMINATION
C
 9000 CONTINUE
      TIM2=SECOND(0)-TIM2
      RESULTS(2)=VAL1
      RESULTS(1)=ASNKTS
      RESULTS(5)=VAL2
      RESULTS(6)=[illegible]
      RESULTS(7)=[illegible]
      RESULTS(4)=NIT
      RESULTS(8)=DIT
      RESULTS(10)=VAL3
      RESULTS(3)=TIM1*1000
      RESULTS([illegible])=TIM2*1000
      RETURN
      END
```

B-7

Chromatics (BASIC)
Cobb County Data Development
Program for
Dial-A-Ride

```
10 REM ***** PROGRAM TRIPS *****
20 REM ***
30 REM *** THIS PROGRAM DISPLAYS A MAP OF COBB COUNTY GEORGIA
40 REM *** IT CAN BE USED TO CREATE OR ADD TO A TEXT FILE
50 REM ***
60 DATA 425,30,505,90
70 DATA 425,170,505,210
80 DATA 425,250,505,300
90 DATA 425,330,505,390
100 DATA 425,430,505,490
110 FILE0=0
120 DIM LABEL$(5),PLACE(5)
130 LABEL$(1)="ADD COMP"
140 PLACE(1)=8
150 LABEL$(2)="REDRAW MAP"
160 PLACE(2)=1
170 LABEL$(3)="SAVE TRIP"
180 PLACE(3)=5
190 LABEL$(4)="OLD FILE"
200 PLACE(4)=8
210 LABEL$(5)="NEW FILE"
220 PLACE(5)=8
230 REM *****MENU PLACE*****
240 FOR I=1 TO 5
250 FOR J=1 TO 4
260 READ MENU(I,J)
270 NEXT J
280 NEXT I
290 PRINT CHR$(12);
300 REM *****DRAW MENU AND MAP*****
310 PRINT "~G~P~K~C7 + 2,2-510,510";
320 PRINT "+ 400,2,400,510";
330 PRINT "~C5~U";
340 PLOT 100,490
350 PRINT CHR$(21);
360 PRINT "COBB COUNTY GEORGIA"
370 FOR I=1 TO 5
380 PRINT "~G~P~C";
390 PRINT USING "#";I;
400 PRINT "+";
410 PLOT MENU(I,1),MENU(I,2),MENU(I,3),MENU(I,4)
420 PRINT "~C7~L +";
430 PLOT MENU(I,1),MENU(I,2),MENU(I,3),MENU(I,4)
440 PRINT "~G~L~C";
450 PRINT USING "#";I;
460 PLOT MENU(I,1)+PLACE(I),MENU(I,2)-10
470 PRINT CHR$(21);LABEL$(I);
480 NEXT I
490 PRINT "~G~K~C7+";
500 PLOT 250,2,400,90
510 DOS"DRAW MAP"
520 PRINT "~G~P";CHR$(27);"U";
530 PRINT CHR$(21);CHR$(7);
540 PRINT CHR$(27);"OA1";"~W 002,003,399,505";
550 PRINT CHR$(27);"OA2";"~W 251,003,399,080";
560 PRINT CHR$(27);"OA3";"~W 401,003,509,509";
570 PRINT CHR$(27);"OA2";"~C7";"SELECT NEW OR OLD FILE";
580 PRINT CHR$(7);CHR$(27);"OA0";
590 REM *****LIGHT PEN INTERRUPT ROUTINE*****
600 OUT&H00,0
610 ON ERROR #2 GOTO 630
620 GOTO 620
630 REM *****ORIGIN POINT READ INPUT*****
640 X=CURSX(4)
```

```
650 Y=CURSY(4)
660 IF Y > 400 THEN GOTO 940
670 X1=X
680 Y1=Y
690 PRINT "~G~P~C3";
700 PRINT "*";
710 PLOT X1-5,Y1-5,X1+5,Y1+5
720 OUT&H90,0
730 RESUME 740
740 PRINT CHR$(21);
750 ON ERROR #2 GOTO 770
760 GOTO 760
770 REM *****DESTINATION POINT INPUT*****
780 X2=CURSX(4)
790 Y2=CURSY(4)
800 IF Y2 > 400 GOTO 2070
810 PRINT CHR$(27);"OA2";CHR$(12);CHR$(27);"OA0";
820 PRINT "~G~L~C3";
830 PRINT "*";
840 PLOT X2-5,Y2-5,X2+5,Y2+5
850 OUT&H90,0
860 FESUME 870
870 PRINT CHR$(21);
880 PRINT "~G~C3";
890 PRINT "*";
900 PLOT X1,Y1
910 PLOT X2,Y2
920 PRINT CHR$(21);CHR$(7);
930 GOTO 590
940 REM *****MENU LOCATE ROUTINE*****
950 IF X <   5 OR  X > 510 THEN GOTO 1010
960 IF Y >  25 AND Y < 100 THEN GOTO 1030
970 IF Y > 125 AND Y < 200 THEN GOTO 1240
980 IF Y > 225 AND Y < 300 THEN GOTO 1420
990 IF Y > 325 AND Y < 400 THEN GOTO 1590
1000 IF Y > 425 AND Y < 500 THEN GOTO 1920
1010 OUTRH90,0
1020 RESUME 590
1030 REM *****MARK EOF ROUTINE*****
1040 PRINT CHR$(27);"OA3"
1050 PRINT "~G~K~1*";
1060 PLOT MENU(1,1),MENU(1,2),MENU(1,3),MENU(1,4)
1070 PRINT "~C7~2*";
1080 PLOT MENU(1,1),MENU(1,2),MENU(1,3),MENU(1,4)
1090 PRINT CHR$(21);CHR$(27);"OA0";
1100 PRINT #6: CHR$(30);
1110 DOS"CLOSE 6 "+NAME$
1120 IF FILEO = 1 THEN DOS"CL(SE 5"
1130 FILEO=0
1140 PRINT CHR$(7);CHR$(27);"OA0";CHR$(12);
1150 PRINT CHR$(21);"~C7";"DO YOU WANT TO CREATE "
1160 PRINT " OR ADD TO ANOTHER FILE"
1170 INPUT ANS$
1180 IF ANS$="NO" THEN STOP
1190 PRINT " "
1200 PRINT "SELECT NEW OR OLD FILE"
1210 PRINT CHR$(7);CHR$(27);"OA0"
1220 OUT&H90,0
1230 RESUME 590
1240 REM *****[illegible] MAP ROUTINE*****
1250 PRINT CHR$(27);"OA3";CHR$(12);CHR$(27);"OA0";
1260 PRINT "~G~C3~[illegible]";
1270 PLOT 101,480
1280 PRINT CHR$(21);
```

```
[illegible] PRINT "CCO[illegible] STATION"
1300 PRINT "^C[illegible]";
1310 PLOT [illegible](2,1),[illegible](2,3),[illegible](2,4)
1320 PRINT "^G[illegible]";
1330 PLOT [illegible](2,[illegible]),[illegible](2,2),[illegible](2,3),[illegible](2,4)
1340 PRINT CHR$(2[illegible]);
1350 PRINT "^G[illegible]";
1360 PLOT 2[illegible]0,[illegible]
1370 DOS"OPEN [illegible]"
1380 PRINT "[illegible]";
1390 PRINT CHR$(2[illegible])CHR$(7);
1400 OUT&H50,0
1410 RESUME
1420 REM ****[illegible] PLOT ROUTINE****
1430 PRINT CHR$(27)"OA2";
1440 PRINT "^G^Z[illegible]";
1450 PLOT MIN[illegible](3,1),MIN[illegible](3,2),MINL(3,3),MINL(3,4)
1460 PRINT #6: USING "###": X1;
1470 PRINT #6: USING "$";",";
1480 PRINT #6: USING "###": Y1;
1490 PRINT #6: USING "$";",";
1500 PRINT #6: USING "###": Y2;
1510 PRINT #6: USING "$";",";
1520 PRINT #6: USING "###": Y2;
1530 PRINT #6: CHR$(13);
1540 PRINT "^G^Z[illegible]";
1550 PLOT [illegible](3,[illegible]),MINM(3,2),MINM(3,3),MCNM(3,4)
1560 PRINT CHR$(7):CHR$(27)"OA[illegible]";
1570 OUT&H50,0
1580 RESUME 590
1590 REM ****OLD FILE ROUTINE****
1600 PRINT CHR$(27)"OA2";CHR$(12);CHR$(7);
1610 PRINT CHR$(2[illegible])"^C";"INPUT OLD FILE NAME"
1620 INPUT NAME$
1630 PRINT CHR$(27)"OA[illegible]";
1640 IF FILEO=1 THEN DOS "CLOSE 5"
1650 FILEO=0
1660 DOS "OPEN 5 R "+NAME$
1670 FILEO=1
1680 DOS "OPEN 6 W 7.DOS"
1690 INPUT #5: X1,Y1,X2,Y2
1700 PRINT #6: USING "###": X1;
1710 PRINT #6: USING "$";",";
1720 PRINT #6: USING "###": Y1;
1730 PRINT #6: USING "$";",";
1740 PRINT #6: USING "###": X2;
1750 PRINT #6: USING "$";",";
1760 PRINT #6: USING "###": Y2;
1770 PRINT #6: CHR$(13);
1780 PRINT "^G^L^OS<";
1790 PLOT X1-5,Y1-5,X1+5,Y1+5
1800 PLOT X2-5,Y2-5,X2+5,Y2+5
1810 PRINT "*";
1820 PLOT X1,Y1
1830 PLOT X2,Y2
1840 PRINT CHR$(21);
1850 IF NOT EOF(5) THEN GOTO 1690
1860 PRINT CHR$(27)"OA2";
1870 PRINT "[illegible]"
1880 PRINT CHR$(2[illegible])"^C";"[illegible]GIN ENTERING TRIPS"
1890 PRINT CHR$(7);CHR$(27)"O[illegible]0";
1900 OUT&H50,0
1910 RESUME 590
1920 REM ****NEW FILE ROUTINE****
```

```
[illegible] PRINT CHR$(27) [illegible] CHR$(27)[illegible];
[illegible] PRINT CHR$(27) [illegible] NOT FILE [illegible]
[illegible]
[illegible] DEF [illegible]
[illegible] PRINT " "
[illegible] PRINT "[illegible] THE [illegible]"
[illegible] PRINT CHR$(27)
[illegible] PRINT CHR$(27)[illegible]
2010 POKE [illegible]
2020 [illegible]
2030 REM [illegible] DESTINATION OUT OF RANGE ROUTINE[illegible]
2040 PRINT CHR$(27)[illegible] CHR$(27)[illegible]
2050 PRINT CHR$(27)[illegible] DESTINATION OUT OF RANGE"
2060 PRINT " "
2070 PRINT "[illegible]"
2080 PRINT CHR$(27)[illegible]
2090 OUT [illegible]
2100 RETURN [illegible]
```

B-8

Chromatics (BASIC)
Stand Alone Programs for
Dial-A-Ride

```
[illegible] CLEAR [illegible]
[illegible] C[illegible]=0
[illegible] PRINT "[illegible]"
[illegible] GOSUB [illegible]
[illegible] [illegible]=[illegible]
[illegible] GOSUB [illegible]
[illegible] [illegible]
[illegible] ON [illegible] GOTO [illegible]
[illegible] [illegible]
[illegible] GOTO [illegible]
[illegible] REM *** MAIN [illegible] ROUTINE
[illegible] [illegible](1)[illegible](1)
[illegible] GOSUB 4[illegible]
5[illegible] IF [illegible] THEN RESUME 150
520 F1=[illegible]
540 F2=6
550 IF F1=6 THEN GOSUB 1000: RESUME 140
560 PRINT "~N1,4813481[illegible]";CHR$(12);
562 H=0
564 GOSUB 4500
570 H=F1+1
580 GOSUB 4500
590 IF F1=1 THEN GOTO 5900
592 IF F1=2 THEN GOTO 6900
594 IF F1=3 THEN GOTO 7900
595 OUT&H83,0
600 RESUME
2000 REM *** TRANSMISSION ROUTINES
2100 REM *** XMIT/INIT
2110 ON ERROR GOTO 0
2199 RETURN
2200 REM *** XMIT/COMPLETE
2210 PRINT CHR$(27);"CAF";CHR$(27);"IA4";
2220 PRINT CHR$(27);"CA0";CHR$(27);"IA0";
2230 RETURN
2300 REM *** RECEIVE/INIT
2310 ON ERROR GOTO 0
2320 PRINT CHR$(27);"CAF";CHR$(27);"IA4";
2330 Z=0   ' THIS IS A DUMMY STATMENT
2340 PRINT CHR$(27);"IA0";
2350 RETURN
2400 REM *** RECEIVE/COMPLETE
2410 PRINT CHR$(27);"IA4";
2420 PRINT CHR$(27);"IA0";CHR$(27);"CA0";
2430 RETURN
3000 REM *** INITIALIZE
3010 DOS"OPEN 5 R NETSAVE"
3020 INPUT #5: NT
3030 DIM TP(NT,4),TC(NT,2),TA(NT),CK(20)
3040 FOR I=1 TO NT
3050 INPUT #5:TP(I,1),TP(I,2),TP(I,3),TP(I,4),TC(I,1)
3060 TC(I,2)=TC(I,1)
3070 NEXT I
3080 DOS"CLOSE 5"
3090 PRINT "~H0,0,511511~;3";CHR$(12);"~IF";
3095 GOSUB 5100
3100 RETURN
3200 REM *** MACRO - LOCATE TRIP
3210 TN=0:NT=0
3220 FOR I=1 TO NT
3230 IF ABS(X-TP(I,1))<9 AND ABS(Y-TP(I,2))<9 THEN NT=I:GOTO 3250
3240 IF ABS(X-TP(I,3))>6 OR ABS(Y-TP(I,4))>6 THEN [illegible]
[illegible] NT=-1
[illegible] TN=I:PRINT CHR$(7);:RETURN
```

```
3260 NEXT I
3270 RETURN
3400 REM *** MACRO - DRAW TRIP (TRIP I, COLOR C)
3410 PRINT "~C";
3420 PRINT USING "#";C;
3430 PRINT "~F*";
3440 PLOT TP(I,1)-5,TP(I,2)-5,TP(I,3)+5,TP(I,2)+5
3450 PRINT "~L*";
3460 PLOT TP(I,3)-5,TP(I,4)-5,TP(I,3)+5,TP(I,4)+5
3470 PRINT "*";
3480 PLOT TP(I,1),TP(I,2),TP(I,3),TP(I,4)
3490 RETURN
3600 REM *** MACRO - DRAW CLUSTER (CLUSTER CN, COLOR C)
3605 AT=0
3610 PRINT "~C";
3620 PRINT USING "#";C;
3630 FOR I=1 TO NT
3640 IF TC(I,2)<>CN THEN 3670
3655 IF C=0 THEN TA(I)=-1
3660 IF TA(I)<>0 THEN GOSUB 3400:AT=AT-1
3665 IF TA(I)=-1 THEN TA(I)=0
3670 NEXT I
3680 RETURN
3800 REM *** MACRO - LOCATE CLUSTER
3810 CN=0
3820 IF Y>400 THEN GOTO 3900
3830 GOSUB 3200
3840 IF TN=0 THEN RETURN
3850 CN=TC(TN,2)
3855 C=TC(TN,1)
3860 RETURN
3900 REM LOCATE CLUSTER FROM COLOR
3910 K=K+1
3920 IF K>NT THEN K=0:MJ=0:RETURN
3930 IF TC(K,1)<>C THEN 3910
3935 IF MJ=0 THEN 3970
3940 FOR J=1 TO MJ
3950 IF TC(K,2)=CK(J) THEN GOTO 3910
3960 NEXT J
3970 MJ=MJ+1:CK(MJ)=TC(K,2)
3980 CN=TC(K,2)
3990 C=TC(K,1)
3995 RETURN
4000 REM *** MACRO - LOCATE ITEM ON MENU (MENU M)
4010 MI=0
4020 FOR I=1 TO MX(M)
4030 IF X<MENU(M,I,1) OR X>MENU(M,I,3) THEN 4060
4040 IF Y<MENU(M,I,2) OR Y>MENU(M,I,4) THEN 4060
4050 GOTO 4080
4060 NEXT I
4070 RETURN
4080 MI=I:PRINT CHR$(7)
4082 IF M<>0 THEN 4088
4084 PRINT "~L~W350400511511";CHR$(12);"~F";
4086 GOTO 4090
4088 PRINT "~L~W1,400350511";CHR$(12);"~F";
4090 PRINT "~L~C7~I";
4100 PLOT MENU(M,I,1),MENU(M,I,2),MENU(M,I,3),MENU(M,I,4)
4110 PRINT "~2";
4120 RETURN
4200 REM *** COLOR MENU READ-IN ROUTINE
4205 PRINT CHR$(12)
4210 PRINT"ENTER S R NCOUN"
4220 INPUT A$: M
```

```
4330 DIM MENU(MM,[illegible],5),M[illegible],HOMENU([illegible],15,3),[illegible],SMX([illegible])
4340 FOR K=0 TO MM
4350 INPUT #5; MX(K),SMX(K)
4360 XX=MX(K)
4370 FOR I=1 TO XX
4380 INPUT #5; MENU(K,I,1),MENU(K,I,2),MENU(K,I,3),MENU(K,I,4),MENU(K,I,5)
4390 NEXT I
4400 XX=SMX(K)
4410 IF XX = 0 THEN 4460
4420 FOR I = 1 TO XX
4430 INPUT #5; HOMENU(K,I,1),HOMENU(K,I,2),HOMENU(K,I,3),[illegible](K,I)
4440 NEXT I
4450 NEXT K
4470 DOS"CLOSE 5"
4480 RETURN
4500 REM *** MENU DRAW ROUTINE
4510 PRINT "~G~K";
4520 FOR I=1 TO MX(M)
4530 PRINT "~F~C";
4540 PRINT USING "&";MENU(M,I,5);
4550 PRINT "+";
4560 PLOT MENU(M,I,1),MENU(M,I,2),MENU(M,I,3),MENU(M,I,4)
4570 PRINT "~G7~L+";
4580 PLOT MENU(M,I,1),MENU(M,I,2),MENU(M,I,3),MENU(M,I,4)
4590 NEXT I
4600 PRINT CHR$(21);
4610 IF SMX(M)=0 THEN RETURN
4620 FOR I = 1 TO SMX(M)
4630 PRINT "~G~U";
4640 PLOT HOMENU(M,I,1),HOMENU(M,I,2)
4650 PRINT "~C";
4660 PRINT USING "&"; HOMENU(M,I,3);
4670 PRINT CHR$(21);[illegible](M,I);
4680 NEXT I
4685 PRINT "~G";
4690 RETURN
5000 REM ***** EDIT FUNCTIONS
5020 IF Y<400 THEN 5060
5030 IF X<350 THEN K=0:NJ=0:PRINT "~I~H350400511511";CHR$(12);"~;F";
5040 IF X<350 THEN M=2:GOSUB 4000:F2=N
5050 IF X>350 THEN M=0:GOSUB 4000:C=[illegible]
5060 ON F2 GOTO 5100,5400,5200,5500
5070 IF F2>0 THEN 5900
5080 PRINT "~W1,401510510";CH[illegible](12);"~C7*350400500511";
5085 PRINT "~W1,1,510400~;6";CHR$(12);"~;F";
5090 F2=0:F1=0:OUT&H90,0: RESUME 120
5100 REM *** EDIT - RESTORE
5110 PRINT "~W1,1,510399~;F";CHR$(12);
5120 FOR I=1 TO NT
5130 TA(I)=1
5140 C=TC(I,3)
5150 GOSUB 5400
5160 NEXT I
5170 PRINT "~W1,1,511511~;A";CHR$(12);"~;F";
5180 F2=0
5185 IF F1=4 THEN RETURN
5190 GOTO 5000
5200 REM *** EDIT - SUBDUE ARC
5210 IF Y<400 THEN 5230
5215 PRINT "~;A~W1,1,511400";CHR$(12);"~;F";
5220 GOTO 5900
5230 PRINT "~;B~W1,1,511400";CHR$(12);"~;F";
5240 GOSUB 5800
5250 IF TA=0 THEN 5900
```

```
5260 IA(TA)=0
5270 C=0:T=TA
5280 GOSUB 5400
5290 GOTO 5900
5400 REM *** EDIT - SUBTLE CLUSTER
5405 IF Y<400 OR X>350 THEN 5500
5410 K=0
5420 PRINT "~:8~W1,1,511400";CHR$(12);"~:F";
5440 GOTO 5900
5500 GOSUB 3000
5510 IF CN=0 THEN 5900
5520 C=0
5530 GOSUB 3600
5540 GOTO 5900
5600 REM *** EDIT - HIGHLIGHT CLUSTER
5610 IF Y<400 OR X>350 THEN 5700
5615 K=0
5620 PRINT "~:6~W1,1,511400";CHR$(12);"~:F";
5625 MJ=0
5630 GOTO 5900
5700 GOSUB 3000
5710 IF CN=0 THEN 5900
5720 PRINT "~:8~W1,1,511400";CHR$(12);"~:F~1";
5730 GOSUB 3600
5740 PRINT "~2";
5750 GOTO 5900
5900 REM *** EDIT - RESET LIGHT PEN AND RECALL EDIT
5905 OUT&H90,0
5910 RESUME 5920
5920 ON ERROR#2 GOTO 5950
5930 OUT&H90,0
5940 GOTO 5940
5950 X=CURSX(4):Y=CURSY(4)
5960 GOTO 5000
6000 REM **** CLUSTER FUNCTIONS
6010 IF Y<400 THEN 6050
6020 IF X<350 THEN K=0:MJ=0:PRINT "~:8~W0,0,511511";CHR$(12);"~:F";
6030 IF X<350 THEN M=3:GOSUB 4000:F2=MI
6040 IF X>350 THEN M=0:GOSUB 4000:O=MI+1
6050 ON F2 GOTO 6200,6300,6400,6500,6600,6700
6060 IF F2=0 THEN 6900
6070 PRINT "~W1,400,510510";CHR$(12);"~C7~W0,400,350511";
6080 PRINT "~W1,1,510400~:6";CHR$(12);"~:F";
6090 F2=0:F1=0:OUT&H90,0:RESUME 120
6200 REM *** CLUSTER - SELECT CLUSTER
6210 IF Y < 400 OR X>350 THEN 6250
6220 K=0:MJ=0:CC=0:OA=0
6230 PRINT "~:6~W1,1,511400";CHR$(12);"~W350400511511";CHR$(12);"~:F";
6240 GOTO 6900
6250 GOSUB 3000
6260 IF CN=0 THEN 6900
6265 OA=CN:CC=0
6270 PRINT "~:6~W1,1,511400";CHR$(12);"~:F~1";
6280 GOSUB 3600
6290 PRINT "~2";
6295 GOTO 6900
6300 REM *** CLUSTER - CREATE CLUSTER
6310 IF Y<400 THEN 6900
6320 IF X>350 THEN 6350
6330 GOTO 6900
6350 PRINT "~:6~W1,1,511400";CHR$(12);"~:F";
6360 CC=0:CA=CM+1:CM=CM+1
6370 GOTO 6900
6400 REM *** CLUSTER - ADD TO CLUSTER
```

```
6415 IF OC=0 THEN PRINT "~#5~W1,410490~11";CHR$(12);"~:F";:F2=0:GOTO 6900
6417 IF Y<400 THEN 6430
6418 PRINT "~#[illegible]05110[illegible]";CHR$(12);"~:F";
6420 GOTO 6900
6430 GOSUB 3200
6440 IF TN=0 THEN 6900
6450 TC(TN,1)=C0:TC(TN,2)=0A
6460 T[illegible]=C[illegible]=0A
6465 PRINT "~1";
6470 GOSUB 3400
6475 PRINT "~2";
6480 GOTO 6900
6500 REM *** CLUSTER - DELETE FROM CLUSTER
6510 IF OC=0 THEN PRINT "~#5~W1,400490511";CHR$(12);"~:F";:F2=0:GOTO 6900
6520 IF Y<400 THEN 6540
6525 PRINT "~#~W350400511511";CHR$(12);"~:F";
6530 GOTO 6900
6540 GOSUB 3200
6550 IF TN=0 THEN 6900
6555 IF TC(TN,2)<>0A THEN 6900
6560 TC(TN,1)=7:TC(TN,2)=9999:I=TN:C=7
6570 GOSUB 3400
6580 GOTO 6900
6600 REM *** REQUEST CLUSTER INFO
6700 REM *** TRANSMIT CLUSTER INFO
6701 GOTO 6900
6900 REM *** CLUSTER - RESET LIGHT PEN AND RESUME CLUSTER
6910 OUT&H90,0
6920 RESUME 6930
6930 ON ERROR 2 GOTO 6960
6940 OUT&H90,0
6950 GOTO 6950
6960 X=CURSX(0):Y=CURSY(0)
6970 GOTO 6000
7000 REM **** ROUTE FUNCTIONS
7010 IF Y<400 THEN 7200
7020 IF X<350 THEN K=0:MJ=0:PRINT "~#0~W350420511511";CHR$(12);"~F";
7030 IF X<350 THEN M=4:GOSUB 4000:F2=MI
7035 IF X>350 THEN M=0:GOSUB 4000:C=MI-1
7040 ON F2 GOTO 7200,6900,7900,7900
7050 IF F2=0 THEN 7900
7060 PRINT "~W1,401510510";CHR$(12);"~C7~G*350400350511";
7065 GOSUB 7300
7070 PRINT "~W1,1,510400~#0";CHR$(12);"~:F";
7080 F2=0:F1=0:OUT&H90,0: RESUME 120
7200 REM *** ROUTE - SELECT CLUSTER
7210 IF Y<400 OR X>350 THEN 7250
7220 K=0:MJ=0
7225 IF LR=1 THEN GOSUB 7300
7230 PRINT "~#0~W1,1,511400";CHR$(12);"~:F";
7240 GOTO 7900
7250 GOSUB 3800
7260 IF CN=0 THEN 7900
7270 PRINT "~#0~W1,1,511400";CHR$(12);"~:F~1";
7280 GOSUB 3600
7290 PRINT "~2";
7295 GOTO 7900
7300 REM *** ROUTE/SELECT REDRAW ACTIVE CLUSTERS
7305 PRINT "~W1,1,510399~:F";CHR$(12);
7310 FOR I=1 TO NT
7320 IF T/(I)=0 THEN 7350
7330 C=TC(I,1)
7340 GOSUB 3400
7350 NEXT I
```

```
7360 PRINT ""W1,1,511511"";CHR$(12);"";""
7370 ERTS
7380 RETURN
7400 REM *** ROUTE/ROUTE POST PARTIAL TOUR SETTING
7405 PRINT ""W350140511050";"";CHR$(21)
7410 PRINT ""O7"U350";"";CHR$(21);"TOTAL DISTANCE ="";
7415 PRINT USING "####";TD
7420 PRINT ""G";
7430 RETURN
7900 REM *** ROUTE - RESET LIGHT PEN AND CURSOR EDIT
7910 OUT&H90,0
7920 RESUME 7930
7930 ON ERROR#2 GOTO 7960
7940 OUT&H90,0
7950 GOTO 7950
7960 X=CURSX(4):Y=CURSY(4)
7970 GOTO 7880
8000 REM *** ROUTE - ROUTE MACRO
8010 IF CR=0 THEN 7900
8020 PRINT ""W1,1,510399";CHR$(12);
8030 C=2:GOSUB 3600
8040 PRINT ""W35140151051O";CHR$(12);:M=5:GOSUB 4500
8050 RMX=2*AT
8060 DIM RT(2*AT)
8070 LA=0:RB=0:RE=0
8080 LR=1
8100 REM ROUTE/ROUTE RESET LIGHT PEN
8110 OUT&H90,0
8115 RESUME 8120
8120 ON ERROR#2 GOTO 8145
8130 OUT&H90,0
8140 GOTO 8140
8145 X=CURSX(4):Y=CURSY(4)
8150 IF Y<400 THEN 8500
8160 IF X<350 THEN 8100
8170 M=5:GOSUB 4000:F3=MI
8180 ON F3 GOTO 8200,8300,8700
8190 GOTO 8100
8200 REM ROUTE/ROUTE START
8210 RB=-1:TD=0
8220 LA=0:PRINT "";4"W1,1,510,399";CHR$(12);
8230 GOTO 8100
8300 REM ROUTE/ROUTE BACK-UP
8310 PRINT ""E"W350400511511";CHR$(12);"";"";
8320 IF LA<2 THEN 8100
8330 A1=ABS(RT(LA)):H1=A1/RT(LA)
8340 A2=ABS(RT(LA-1)):H2=A2/RT(LA-1)
8350 IF H1>0 THEN X1=TP(A1,1):Y1=TP(A1,2)
8360 IF H1<0 THEN X1=TP(A1,3):Y1=TP(A1,4)
8370 IF H2>0 THEN X2=TP(A2,1):Y2=TP(A2,2)
8380 IF H2<0 THEN X2=TP(A2,3):Y2=TP(A2,4)
8390 PRINT ""CE"C"";:PLOT X1,Y1,X2,Y2:PRINT ""F*";:PLOT X1,Y1:PRINT "005"L"F";
8392 TD=TD-SQR((X1-X2)^2+(Y1-Y2)^2)
8393 GOSUB 7000
8395 LA=LA-1:GOTO 8695
8400 REM *** ROUTE/ROUTE STARTING POINT GIVEN
8410 LA=0
8430 GOSUB 3200
8440 IF TN=0 THEN 8100
8450 IF TC(TN,2)<>CN OR TA(TN)=0 THEN 8100
8460 IF MF<1 THEN 8100
8470 LA=1:RT(1)=TN:RB=0
8475 PRINT ""E"W350400511511";CHR$(12);
8480 PRINT ""C"W1,1,511400";CHR$(12);
```

```
8490 TD=0
8495 X2=TP(TN,1):Y2=TP(TN,2):GOTO 8655
8500 REM *** ROUTE/ROUTE NEXT POINT
8510 IF FP=-1 THEN 8400
8520 IF LA=0 THEN 8190
8540 GOSUB 3200
8550 IF TN=0 THEN 8100
8560 IF TC(TN,2) <> CN OR TA(TN)=0 THEN 1100
8562 PRINT "":[illegible](12)[illegible]
8565 FG=0
8570 FOR J=1 TO LA
8580 IF RT(J)=TN*HT THEN 8100
8590 IF RT(J)=-TN*HT THEN FG=1
8600 NEXT J
8605 IF FG=0 AND HT<0 THEN 8100
8610 LA=LA+1
8620 RT(LA)=TN*HT
8625 A1=ABS(RT(LA-1)):H1=A1/RT(LA-1)
8630 IF H1>0 THEN X1=TP(A1,1):Y1=TP(A1,2)
8635 IF H1<0 THEN X1=TP(A1,3):Y1=TP(A1,4)
8640 IF HT>0 THEN X2=TP(TN,1):Y2=TP(TN,2)
8645 IF HT<0 THEN X2=TP(TN,3):Y2=TP(TN,4)
8647 TD=TD+SQR((X1-X2)^2+(Y1-Y2)^2)
8650 PRINT "":C^C4"::PLOT X1,Y1,X2,Y2
8655 PRINT "":C^C4^1^F<"::PLOT X2,Y2:PRINT "[illegible]";
8658 GOSUB 7600
8660 GOTO 8100
8700 REM ROUTE/ROUTE STOP
8705 PRINT "":6^H350400511511":CHR$(12);"":F";
8710 IF LA<2*PRT THEN 8100
8720 PRINT ""H3514035105120":CHR$(12)::N=1:GOSUB 4500
8722 PRINT ""G";
8725 ERASE RT
8730 GOTO 7900
```

```
#,      C3DIAL-A-RIDE PRIMARY MENU FILE C2
#,0,     C4COLOR MENU C2
360,460,385,500,0
395,460,420,500,1
430,460,455,500,2
465,460,490,500,3
360,410,385,450,4
395,410,420,450,5
430,410,455,450,6
465,410,490,450,7
#,5,     C4CENTRAL MENU C2
5,430,45,480,7
55,430,95,480,7
105,430,145,480,7
155,430,195,480,7
14,425,2,"EDIT"
53,425,4,"CLUSTER"
110,425,6,"ROUTE"
150,425,3,"INITIATE"
70,502,7,"DIAL-A-RIDE PRINCIPAL SELECTIONS"
#,9,     C4EDIT MENU C2
5,430,45,480,7
55,430,95,480,7
105,430,145,480,7
155,430,195,480,7
205,430,245,480,7
6,425,2,"RESTORE"
58,425,4,"IGNORE"
58,414,4,"CLUSTR"
107,425,6,"IGNORE"
115,414,6,"APC"
```

```
151,425,3,"SELECT"
155,414,3,"CLUSTR"
210,425,7,"EXIT"
70,502,7,"DIAL-A-RIDE EDIT SUBFUNCTION"
#,10,    CRLLSION MENU C2
5,430,40,480,7
50,430,85,480,7
95,430,130,480,7
140,430,175,480,7
185,430,220,480,7
230,430,265,480,7
275,430,310,480,7
5,425,6,"SELECT"
5,414,6,"CLUSTR"
50,425,3,"CREATE"
50,414,3,"CLUSTR"
95,425,2,"ADD TO"
95,414,2,"CLUSTR"
138,425,4,"DEL FRM"
142,414,4,"CLUSTR"
190,425,5,"REQST"
236,425,6,"XMIT"
279,425,7,"EXIT"
70,502,7,"DIAL-A-RIDE CLUSTER SUBFUNCTION"
#,9,    CARCUTING MENU C2
5,430,45,480,7
55,430,95,480,7
105,430,145,480,7
155,430,195,480,7
205,430,245,480,7
5,425,3,"SELECT"
5,414,3,"CLUSTR"
```

```
54,420,2,"ROUTE"
100,400,5,"[illegible]"
102,414,5,"ROUTING"
180,425,6,"EXIT"
150,414,6,"ROUTING"
210,425,7,"EXIT"
70,532,7,"DIAL-A-RIDE ROUTING SUBFUNCTION"
8,3,   C4ROUTING SUBMENU C2
370,430,400,460,7
410,430,440,460,7
450,430,480,460,7
370,414,2,"START"
415,414,8,"BACK"
455,414,4,"STOP"
```