DEVELOPMENT OF A
TRANSIT SERVICE PLANNING MODEL

by

JOHN A. DETORE, JR.

B.S. THE MASSACHUSETTS INSTITUTE OF TECHNOLOGY
(1980)

Submitted to the Department of
Civil Engineering in Partial
Fulfillment of the Requirements
for the Degree of

MASTER OF SCIENCE IN TRANSPORTATION

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May, 1983

© Massachusetts Institute of Technology

Signature of Author

Department of Civil Engineering, May 13, 1983

Certified by

George A. Kocur, Thesis Supervisor

Accepted by

Amedeo Odoni, Chairman, MST Program Committee

Accepted by

Francois Morel, Chairman, Departmental Committee

OCT 21 1983 Archives
DEVELOPMENT OF A
TRANSIT SERVICE PLANNING MODEL

by

JOHN A. DETORE, JR.

Submitted to the Department of Civil Engineering
on May 13, 1983 in partial fulfillment of the
requirements for the Master of Science in
Transportation

ABSTRACT

This thesis describes the design and implementation of FRACAS, a microcomputer
software package aimed at helping transit managers and planners with service and
strategic planning. This interactive, optimizing model improves the capabilities
of transit properties to do strategic planning, while emphasizing the human
component in optimization. Using readily available data, FRACAS outputs the fare,
service level, and projected ridership that best meets system objectives.

Several conclusions are made throughout the development of the thesis that become
the premise for the design concept of FRACAS. First, given a future of financial
austerity, there will be a need for more strategic planning in transit agencies.
Second, to be successful, any strategic planning model introduced must be
inexpensive to purchase, run, and maintain. Third, given that managers as well as
planners would be the ultimate users of such a system, the model must be
"user-friendly," easy to learn, and simple to execute repetitively in order to do
sensitivity analyses and to build user intuition.

Substantial effort was devoted to the programming of FRACAS onto an Apple II
computer. Although the system now working meets our objective of minimal expense
to the end user, in hindsight it may have been prudent to use a more powerful
machine.

After the system was programmed and debugged, we turned our attention to
implementation problems. Traditionally, the strategic planning process has been a
political one, and the widespread acceptance of FRACAS would mean radical change.
In addition, established planners at transit properties may reject the model, for
"how can this box understand the 'unquantifiable' qualities of their particular
municipality?" We conclude that for FRACAS to succeed, it will have to be viewed
as a tool for increasing the efficiency of planners—a tool meant to interact with
and aid planners' judgment rather than supplant it.

Thesis Supervisor: George A. Kocur
Title: Assistant Professor of Civil Engineering

2
ACKNOWLEDGEMENTS

I would like to take this opportunity to thank those who not only helped me with this project, but those who have made my educational experiences at M.I.T possible.

I heartily thank all those at Dennis Elliott and Associates for providing so much support and encouragement throughout the year. I am especially indebted to them for the use of their Apple II system -- the machine has been put to good use.

Professors Michael Meyer and Nigel Wilson were very helpful in the formulation of my fifth chapter on the implementation of FRACAS into actual transit system use. I am especially indebted to Mike for giving time out of his very tight schedule, and for the support and friendship he has given me for four of my five years at M.I.T.

To Professor George Kocur, my thesis advisor: I have learned so much this year. Thank you for your constant attention, unfailing support, and most recently, reviewing multiple chapters of my thesis every night. Thank you for introducing me to real computer programming, and for letting me in on a Master's project that I can be proud of.

For my parents, I wish I could return all the love and support you have given me. Getting an education at M.I.T. is not easy, and you are the most important reason I have been able to succeed.
# TABLE OF CONTENTS

1. Introduction and Summary ........................................... 6  
   1.1 Overview .................................................................. 6  
   1.2 Perspective ........................................................... 7  
   1.3 Methodological Issues ............................................. 8  
   1.4 Implementation .................................................... 11  

2. Review of Other Models ................................................ 13  
   2.1 BUSHMODEL and STRATEGEM ................................ 13  
   2.2 TOP .................................................................. 14  

3. User Interaction .......................................................... 17  
   3.1 Overview .............................................................. 17  
   3.2 Input and Output Quantities .................................... 17  
   3.3 Implementation of Interactive Screens ..................... 21  

4. System Development ..................................................... 32  
   4.1 Introduction .......................................................... 32  
   4.2 Hardware and Software Selection ............................. 32  
   4.3 Implementation of User Interface ......................... 34  
   4.4 FRACAS System Structure ..................................... 37  
   4.5 Programming Considerations ................................. 39  

5. Implementation .......................................................... 44  
   5.1 User Testing .......................................................... 44  
   5.2 The Need for FRACAS ......................................... 45  
   5.3 Perceived Implementation Problems ....................... 46  

Appendix A: FRACAS User's Guide .................................... 51  
   A.1 Introduction .......................................................... 51  
   A.2 Program Structure ................................................ 54  
   A.3 Running FRACAS .................................................. 57  
   A.4 The Diskettes ...................................................... 61  
   A.5 An Example Run .................................................... 63  

References ................................................................. 86
TABLE OF FIGURES

1. The Objective Screen .............................................. 25
2. The Constraints Screen ........................................ 27
3. The Operating Screen ........................................... 28
4. The Existing Screen .............................................. 30
5. The Market Screen ................................................. 30a
6. Program Tasks ..................................................... 38
A1. Program Structure ............................................ 55
A2. Example Objective Screen ..................................... 69
A3. Example Constraints Screen ................................ 71
A4. Example Area Map ............................................... 72
A5. Example Operating Screen ................................... 75
A6. Example Existing Screen ..................................... 78
A7. Sensitivity Charts ............................................... 80
A8. Example Market Screen ..................................... 83
A9. Example Output .................................................. 84
SECTION ONE
INTRODUCTION AND SUMMARY

1.1 Overview

This thesis describes the design and implementation of FRACAS, a microcomputer software package aimed at helping transit managers and planners with service and strategic planning. This interactive, optimizing model improves the capabilities of transit properties to do strategic planning, while emphasizing the human component in optimization. It is an initial step in merging the traditionally distinct computer roles of optimization systems, which deliver a "best" answer for a given problem, and decision support systems, which emphasize the processing of databases into a form easily interpretable by managers. When used by experienced planners and managers, the model will aid in building an intuition and understanding of the transit system, as well as delivering an approximate optimum answer with which to base strategic planning.

For a given transit system corridor or route, FRACAS specifies a combination of service and fare that best achieves system objectives. The service is defined by the number of routes in the analysis (in corridor and system wide), their average length, the average headway and the load per bus. Express and local service, and peak and off-peak time periods can be treated jointly. FRACAS computes the service levels, ridership, revenues and costs of options, and provides statistics on bus-miles, bus-hours, and passenger-miles, for example.
This section reviews the current environment that transit planning is in and how that environment affects planning decisions. We then examine methodological issues from the perspective of what a planning tool should do to be responsive to the issues facing transit today. Finally, the way FRACAS addresses these methodological issues is presented.

1.2 Perspective

The direction that computerized planning technology should take must be understood in the context of the environment that transit now faces and will face in the future. For example, it should always be kept in mind that resources are limited, and and transit agencies will generally require that a computer system not be expensive to purchase, operate, and not require extensive additional data collection.

It is also true that transit management has to support goals that extend beyond meeting this year's budget, though most effort of the typical property is expended on current operations. As an institution, the management of an agency must become responsive to organizations that have an effect on the long term success of the agency. Thus a planning staff needs not only to be able to handle short-term decisions concerning the routing and scheduling adjustments, but must be able to address service/subsidy trade-offs with city governments and federal officials.

In fact, our changing political outlook may substantially influence the transit planning activities of the future. President Reagan, for example, has proposed the elimination of operating subsidies. Transit agencies need to be able to
state what services will be lost for what loss in funding, and this mechanism should be quick, responsive and accurate.

A spectrum of activities can thus be conceived from basic "tactical" decisions, such as schedule adjustments, to more "strategic" issues such as fleet size and area serviced. With the increasing volatility of federal funding for operating subsidies, the needs for analysis may extend beyond the typical strategic questions to include the need for a system that provide quick analysis of potential services that can be provided given a proposed budget structure determined by the political process. FRACAS addresses the strategic end of this spectrum, leaving the tactical issues to experienced schedulers and detailed route-level models.

1.3 Methodological Issues

From the analyst's point of view, the transit system has existing routes, fares, equipment and resources. Some variables (e.g., subsidies, area of service) may be specified externally, and other variables, are under the operator's control (i.e., fare, headway) and are used to achieve system goals for ridership, user benefits, or service levels.

The specification of a strategic planning system has to address several basic questions. First, what should be the appropriate amount of data required by the system? Second, what processes (optimization, simulation, financial accounting) will be incorporated into the system? Third, on what scale will the processing be done (hand calculator, micro, mini, mainframe)? Last, how interactive should the system be? Each of these questions will be briefly addressed.
Data requirements heavily influence overall cost. The effect can be illustrated by comparing a calculator sketch planning model that may take an analyst a day to work through, but only requires readily available data at essentially no cost, with large regional network models that require several full time staff members just for database management. We choose a goal in developing FRACAS of using only existing data, rather than designing a system that requires data that may be too expensive or time intensive to collect. FRACAS uses only route-level or corridor-level data, and requires no trip tables, networks, demand models, or other model calibration. All data is typically available within the transit organization.

The multiplicity of trade-offs between optimization, the process of finding a "best" answer with analytical or numerical techniques, and simulation, the process of designing a mathematical model of a real system and studying the performance of the model, are difficult to address briefly. However, there are three key issues which seemed to be central to the design of FRACAS. In optimization, more assumptions and abstractions are made in the translation of reality to a tractable model than if simulation were used. Although simulation offers more flexibility in capturing the elements of a real world system, it requires substantially more data, and complex simulations can produce results which are very difficult to interpret (or even to know to be correct). The last aspect of simulation is that it cannot produce "good" strategies; it relies on the analyst, whose intuition is likely to be limited when considering many variables simultaneously in a detailed setting. Thus, the optimization approach was chosen for FRACAS. The key element in transit strategic planning is to be able to trade-off many variables and develop an understanding of the system. We
are willing to sacrifice some detail. Also, the need to generate "good", new strategic options may be more pleasing in today's environment than to microscopically evaluate small detailed changes.

The third design issue is the scale of equipment to use. The choice of scale of equipment is determined by cost: the cost of implementing a model on a calculator, microcomputer, minicomputer, and mainframe computer must be compared. State-of-the-art calculators, although capable of handling a hundred or more variables, do not have the program space to handle most real problems. In addition, their input and output capabilities are usually insufficient for handling the amount of data required even for a modest system without absorbing a substantial amount of an analyst's time. The dedicated microcomputer is much less expensive than a dedicated minicomputer, and possibly less than monthly charges for computer time on a large machine. Such a machine may be capable of handling the demands of a strategic planning system, since this does not require the same level of detail as a simulation, which would be unlikely to operate well on a microcomputer. The costs of a microcomputer might be matched by sharing a minicomputer with other functions, but this limits the distribution of FRACAS to users that can implement these other functions. The minicomputer also presents some major compatibility issues. The possibility of implementing an optimization program on a micro should be the first option to be explored. This is the decision we made for FRACAS.

This leaves the issue of the level of interactiveness with the user. For most users, the possibility of interacting with a computer, rather than submitting runs to a computer, has come about only in the past decade. This evolution has brought about increases in analyst productivity along with increased
opportunities to use the computer to build an understanding of a complex system. This interactivity comes at a hefty price on mainframe computers, which have to spend much computing time on overhead and handling user screens and interactive keyboards. (Optimization, as a rule, took a datafile and returned an answer. Such programs contributed to the use of "batch" (noninteractive) processing.)

With the widespread use of microcomputers, interactivity has received much more emphasis. With only a single user and an inexpensive machine, the overhead necessary to handle interactivity is much less of an issue. Programs such as spreadsheets became popular, and managers and analysts alike discovered that interactive process not only allows the ability to do runs more quickly, but may also help to build the user's intuition of how a system behaves. FRACAS is intended to improve understanding of trade-offs in transit system management and planning, so an interactive implementation on a microcomputer was selected. The computational capabilities of several popular microcomputers appeared to be perhaps adequate for the task. As it turned out, FRACAS is interactive, but barely fits on the microcomputer on which it was implemented.

1.4 Implementation

FRACAS was implemented on an Apple II plus microcomputer and was written in UCSD Pascal. It uses five interactive screens for the input of the data necessary to specify a run. It produces a four screen report, and the user is free to look through and return to any of these screens. FRACAS calculates these output relatively quickly (about ninety seconds), allowing the user to repetitively return to the interactive screens, change a few data, and see the effect. This
allows the user of FRACAS to perform sensitivity analyses and built a greater
intuition into the relationships between service, fare and ridership.

An Apple II system should cost no more than a few thousand dollars, which should
be affordable for most transit properties. The decision for the purchase of
this hardware becomes more justifiable when considering the availability of the
other software available for the Apple, including spreadsheets, accounting
packages, an database management systems.
SECTION TWO
REVIEW OF OTHER MODELS

2.1 BUSMODEL and STRATAGEN

There are three other transit planning models of which we are aware that have been implemented on microcomputers. These are STRATAGEN and BUSMODEL, designed by Colin Buchanan and Partners and implemented on any microcomputer supporting C/PM with 5 1/4" floppy disks [Buchanan and Partners, 1982], and TOP (Transit Operations Planning model), designed by Mark Turnquist, Armin Heyburg, and Stephen Richie and implemented on an Apple /// computer [Turnquist, Heyburg, and Richie, 1981, 1982].

There are many similarities between the BUSMODEL and STRATAGEN. BUSMODEL, through a list of questions, collects input data on present transit demand, costs, growth trends in transit demand, and fare and service elasticities. The model then multiplies the appropriate elasticities to the changes in fare and service input by the user to obtain a multiplier that describes future demand in terms of present demand.

The major strength of the BUSMODEL system is its ability to segregate demand by geographic areas and into classifications of users and time periods. The model also provides the opportunity to specify fares by distance travelled and user type (adult, child, handicapped...). After projecting future demand, these demand figures are used to determine the costs of the service needed by multiplying the appropriate cost by level of service, and the benefits to each population subgroup by multiplying benefit per passenger by passengers. It is
proposed that the summation of these costs and benefits will lead to a "social cost benefit index."

Although the flexibility in specifying a large number of users, time periods, and zones can be useful, it is unlikely that a planner would have access to the different elasticities for each combination of user type and time period, all of which must be input by the analyst. Thus, this model becomes limited by the availability of data to which the planner has access. Second, it is not clear that this model relieves the planner of the task of forecasting; the program appears only to mask the problem, since it asks the planner to input all components of the forecast. The best use of this system may be as a tool for maintaining a database of disaggregated users and as a source of standardized reports. It would only be of use to planners who wish to approach demand forecasting with a simple forecasting tool, but considerable disaggregation of data, which may or may not be a sound forecasting philosophy.

STRATAGEM appears to be an extension of BUSMODEL. Asking virtually the same questions as BUSMODEL, STRATAGEM will handle multiple future dates at the same time, and provides output for train and metro modes in addition to bus. Now, of course, the user must specify the cost inflation and demand growth rate assumed for each mode, time period, and user type. This model also is limited by the simplicity in prediction process and need for large amounts of data that requires the planner to do most of the work.

2.2 TOP

TOP is a model specifically designed for the short-range planning of one or a
few parallel routes within a system. The model requires considerable route-level data. For a given route, the user specifies the route geometry, including signalized intersections and speed limits; the on-off counts by stop; a flat fare; and cost per vehicle, vehicle-km and vehicle-hour. The model then estimates an origin-destination matrix, and uses either an elasticity or multinomial logit demand model to estimate changes in transit demand. This demand model is iterated with a transit performance model to reach an equilibrium route ridership and running time. When the system reaches equilibrium, the new demand is reported. The model will output a ridership summary, level-of-service summary, operating statistics summary, financial summary, the calculated origin-destination file, or a graphic route load profile.

TOP will model changes in headway, fare, or costs and report the new ridership, level-of-service, operating statistics, and financial information. The model should also be capable of modeling changes in route structure (e.g., dropping a little used stop), but it is not clear how easy such a change is on TOP. TOP addresses the "tactical" end of the tactical/strategic planning spectrum; for this reason, the use of both FRACAS and TOP by the same property could provide a relatively "complete" package, and may offer some synergism. For example, TOP outputs the average vehicle speed, which is an input for FRACAS.

Additionally, operators use spreadsheets and plotting/statistical analysis packages for certain strategic decisions as fleet replacement and ridership trend analysis. Since many of these packages are available on the Apple II, many operators will already have the equipment to run FRACAS, or will have additional incentive to purchase an Apple II.
The examination of these models leads to two conclusions:

1. Microcomputers have been used with varying degrees of success for the development and implementation of transit planning software. The systems seem capable of handling sophisticated algorithms and the data necessary to address planning problems.

2. Especially when considering the current political attitudes on transit operating subsidies, there exists a need to extend the scope of transit planning software from the specific, local, tactical approach taken by the authors of TOP, to the more general, strategic and policy oriented end of the planning spectrum.
SECTION THREE
USER INTERACTION

3.1 Overview

This section describes the process of designing the flow of information between the user and FRACAS. Since one of the goals in designing FRACAS was to produce a system that was "user-friendly", there was a substantial amount of effort expended to organize the input data into intuitive groups, and to present the outputs in an easily interpretable set of tables.

The first part of this section discusses the inputs and outputs required by the model, and describes the process of logically organizing this information. The second part develops an actual specification of the interactive screen for FRACAS.

3.2 Input and Output Quantities

3.2.1 General Design Issues.

In designing FRACAS, several questions had to be answered. First, what variables need to be determined by the system to make it an effective strategic planning tool? Second, what data are readily available? And third, what level of detail should the model cover? In other words, where on the scale between a sketch planning model and a network model should this model lie?

The basic approach to these questions was derived in Ko cur (1982). His model,
derived with analytical optimization techniques, solves for the optimal service
and fare levels using a small set of input variables described in this chapter.
The decision variables are the number of routes in the corridor studied, the
average route length, the average fare, and the average headway.

The model applies only to a transit system consisting of radial routes extending
from the CBD (central business district). The analyst may optimize the system
with respect to one of three objectives: 1) the minimization of deficit; 2) the
maximization of weighted ridership minus deficit; and 3) the maximization of
ridership subject to a deficit constraint. Fares and route structures may be
constrained if desired. It is also possible to specify all the service and fare
variables and use the model only to determine the ridership and calculate the
resulting cost of service, revenue, benefit and deficit. The analyst may
consider peak, offpeak or both peak and offpeak service within the model,
setting constraints (such as equal fare) between the two periods. Likewise,
express and/or local service may be considered.

3.2.2 Input Requirements.

The model internally consists of nine cases, each optimizing the system given
data on what objective is desired, which combinations of local or express
service during the peak or offpeak period are to be analyzed, and whether
service or vehicle loading constraints exist. Thus the data needed for the
specification of a case is:

- The objective.
- Whether each decision variable is constrained to a preset value,
  constrained to be equal in the peak and offpeak periods, or free to
vary.
  o Which combinations of service (express, local) and time periods (peak, off-peak) to analyze.

Data is required of the current transit operations, to establish a base from which to estimate changes in ridership, services and cost. For a corridor analysis, the following data would generally be helpful:

  o Current number of routes.
  o Current route length.
  o Current fare.
  o Current number of bus trips.
  o Current ridership.
  o Current percentage going to and from the CBD (helps determine average trip).
  o Current percentage of passengers moving in the peak direction.
  o Current market share for transit into and out of the CBD.

The analysis requires the user to specify the following data about ridership characteristics and overall market conditions for transit:

  o The average walking speed.
  o The maximum tolerated walk distance.
  o Average peak and off-peak CBD parking costs.
  o The ratio of wait time to the headway.
  o Sensitivities of ridership for service and fare. These relate ridership to fare, running time, walk time and wait time. These sensitivity estimates are needed for each service and time period.

Last, the following operating characteristics are required:

  o The maximum policy headway.
  o The length of the analyzed corridor along typical traveled streets.
  o The width of the corridor at its outer edge.
  o The number of expressways in the corridor.
  o The size of the CBD.
  o The average bus operating speed for each service and time period.
  o The length, in hours per day, of each time period.
  o The fixed costs per day and the operating costs per bus.
  o The maximum number of passengers per bus by service and time period.
The transit system will have almost all of these numbers at hand. The model provides curves and defaults to select the market sensitivities, and the CBD market share is readily obtained from the regional planning agency if not known. Other variables are either known from collected data or can be estimated fairly well from experience. No special data collection efforts are needed to support this model.

3.2.3 Output Reports.

FRACAS calculates 33 different outputs for each service and period. This information is organized into a two screen Management Report, containing the decision variables and the overall financial results, and a two screen Technical Report, containing mostly calculated data. The analyst can study these screens freely -- one can easily return to a screen that has already been viewed.

The outputs provided are:

Service and Fares:

- Number of Routes.
- Average Route length.
- Average Headway.
- Average Fare.

Overall Impacts:

- Load Per Bus.
- Mode Share (CBD).

Daily Impacts:

- Cost.
- Revenue.
- Deficit.
- User Benefit.
- Ridership.
Annual Impacts:

- Cost.
- Revenue.
- Deficit.
- User Benefit.
- Ridership.

Daily Statistics:

- Bus-Miles.
- Bus-Hours.
- Number of Bus-Trips.
- Number of Buses.
- Passengers Per Bus-Mile.
- Passenger-Miles.
- Passenger-Miles Per Bus-Mile.
- Cost Per Passenger.
- Revenue Per Passenger.
- Deficit Per Passenger.
- Benefit Per Passenger.
- Operating Cost.
- Fixed Cost.

Other statistics given include:

- Ratio Revenue/Cost.
- Average Travel Time in Minutes.
- Average Walk Time in Minutes.
- Average Wait Time in Minutes.

3.3 Implementation of Interactive Screens

3.3.1 General Design Issues.

Combining the inputs needed and outputs provided by the model with the requirements of the computerized strategic planning tool, a functional specification of the model was developed, which determined the responses of the input screen at each point throughout the execution of the program. Each screen consists of either a group of data entry locations or presents a menu of model choices. There are five interactive data screens which will be described in the
next section.

The emphasis on "user friendliness" was maintained as much as possible. If this system is to be a successful strategic planning tool, it has to remain sophisticated enough to satisfy the planner while remaining flexible and easy to use for the manager. The speed of the system is important, but this should be measured in the time for the user to do a useful analysis and not simply the time necessary to execute the program.

Five of the screens will in the model divide the data into the four input groups presented in section 3.2.2, and each screen is accessible from a sixth screen, the MAIN menu, so that any data value in the system is only one screen away from any other screen. Each screen has a cursor that can be positioned at any value, so that any value can be seen, moved to, or changed without affecting any other values. The system is designed to keep all the data well organized and immediately accessible. All of the important options available to the user are displayed on the screen, so that the system is easy to learn and requires few references to the user's manual. Both the inputs and the outputs are printable with a simple keystroke for permanent records of the analyses done. The system also has the ability to store and load the input values to disk, so that a machine readable version of previous analyses can be kept. Finally, the system is also able to catalog the datasets without resorting to external filing procedures.

3.3.2 Main Menu Screen and Options.

When FRACAS starts, the analyst is presented with the MAIN menu. From this
menu, any of the interactive data screens can be accessed. These screens will be discussed in the following section. Other options include moving to the STORAGE page, requesting that the system process the input data and display the output, and quitting the system.

The STORAGE page gives the analyst the ability to store the information on each of the interactive screens into one named file on disk. With this feature, all the screens can quickly be reset to the values of a previous session that was stored. When the STORAGE page is selected, a screen appears with a menu of storage options and a catalog of all the files currently on disk. This catalog is kept current throughout all storage and retrieval activity. The options available to the analyst include storing, replacing, loading, and deleting a file, plus printing a list of the datasets on disk. The disk drive to be used for storage and retrieval of data can be specified for the convenience of FRACAS users with more than two floppy drives or a hard disk.

The OUTPUT option will process the data that exists on the input screens. First, data is checked for completeness, then the screen is cleared for the listing of messages concerning the calculations done. Finally, the report screens are shown. After leaving the report screens, the user of FRACAS is returned to the MAIN menu.

The last option on the MAIN menu is the QUIT option. This option brings FRACAS to an orderly halt and provides convenient access to the command level of the Pascal system to copy disks.

3.3.3 Specification of Data Screens and Output Format.
Having specified the interactive screens to be used, the task remains to actually design the screens. The data discussed above is divided into five screens: 1) data on the existing system, 2) operating data, 3) market sensitivities, and 4) and 5) objectives and constraints are broken into two menus. Each of these screens is discussed in turn:

The Objective Screen. The objective screen includes the objective of the analysis, the time periods (peak or offpeak) and services (local or express) that are to be analyzed. The corresponding information is needed for existing data. This screen also gives the user opportunity to declare that the number of routes, route length, or fare will be predetermined, although the predetermined values will actually be given on the constraints screen. In this way, a constraint can be removed without actually having to remove the data from the constraint screen, leaving it intact for later analyses. Since all data values consist of a choice from a menu of items, each space will accept only a single digit integer. Any character selected other than a digit is not permitted. Likewise, any value that is not defined as an option is not allowed. If an error is detected in the input provided, a message is displayed at the bottom of the screen, and the cursor is returned to the location of the error. The user can then correct the error. The screen delineates the major available options for the user at any point. In addition, the name of the current datafile of inputs displayed. An illustration of a completed screen is given in figure 1.

The Constraint Screen. This page again will have a cursor capable of selecting the number of routes, route length, fare, or number of trips as a constraint on the future system. Other data on this screen include the maximum
FIGURE 1
THE OBJECTIVE SCREEN

<table>
<thead>
<tr>
<th>Objectives, Constraints, Time Periods</th>
<th>File: GEORGE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time Periods &amp; Service Types:</strong></td>
<td></td>
</tr>
<tr>
<td>1 ....... Analyze Local Routes?</td>
<td>1 - Peak</td>
</tr>
<tr>
<td>4 ....... Analyze Express Routes?</td>
<td>2 - Offpeak</td>
</tr>
<tr>
<td>1 ....... Existing Local Routes?</td>
<td>3 - Both</td>
</tr>
<tr>
<td>4 ....... Existing Express Routes?</td>
<td>4 - None</td>
</tr>
<tr>
<td><strong>1 ....... Objective</strong></td>
<td></td>
</tr>
<tr>
<td>1 - Minimize Deficit</td>
<td></td>
</tr>
<tr>
<td>2 - Max Weighted Riders-Deficit</td>
<td></td>
</tr>
<tr>
<td>3 - Max Riders w/Deficit Constraint</td>
<td></td>
</tr>
<tr>
<td>4 - All Predetermined</td>
<td></td>
</tr>
<tr>
<td><strong>Constraints:</strong></td>
<td></td>
</tr>
<tr>
<td>1 ....... No. of Routes</td>
<td>1 - Model Chooses, Separate</td>
</tr>
<tr>
<td>1 ....... Route Length</td>
<td>2 - Model Chooses, Equal (2 periods)</td>
</tr>
<tr>
<td>1 ....... Fare</td>
<td>3 - Predetermined</td>
</tr>
</tbody>
</table>

Use <-, -> to move cursor
Cntl-C to accept data
Esc to MAIN menu
Cntl-P to print screen
allowable deficit, used in conjunction with the deficit constraint objective, and the value of a rider, used in conjunction with the "weighted ridership less deficit" objective. The fixed costs per day are also on this screen to allow comparison with the deficit constraint. Bars light up above the required data items.

All locations are capable of accepting real numbers; characters other than a digit or decimal place are rejected. Most of the data will require a five digit field with one decimal place. As before, the major options available to the user are shown on the screen. A illustration of this completed screen is given in figure 2.

**The Operating Screen.** Operating data are input on this screen. These data include the length and width of the analysis area, the number of expressways in the area (for express service), the length of the CBD, the number of analysis days in the year, the ratio of total bus-hours to in-service bus hours, the operating cost per bus per hour for both peak and offpeak (if needed), the length of the peak and offpeak periods, the maximum load per bus, and the average bus speeds. These items all require real data with one decimal place. All of these items are used to calculate service levels and costs for alternative transit operating strategies. The Operating screen is illustrated in figure 3.

**The Existing Screen.** The existing data required includes the number of routes, route length, fare, number of bus trips, ridership, percentage of transit riders destined for the CBD, percentage of riders in the peak direction, and the transit market share of CBD trips. All this data is real
### FIGURE 2
THE CONSTRAINTS SCREEN

<table>
<thead>
<tr>
<th>DATA</th>
<th>Constraints, Objectives</th>
<th>File: GEORGE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Predetermined Values**

<table>
<thead>
<tr>
<th>N</th>
<th>L</th>
<th>F</th>
<th>t</th>
<th>M</th>
<th>Co</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>used this run</th>
<th>Number of Routes</th>
<th>Route Length (mi.)</th>
<th>Fare (cents)</th>
<th>No. of Trips</th>
<th>Max Deficit ($/day)</th>
<th>Fixed Costs ($/day)</th>
</tr>
</thead>
<tbody>
<tr>
<td>peak:</td>
<td>100.0 10.0</td>
<td>10.0 12.0</td>
<td>50.0 175.0</td>
<td>300.0 40.0</td>
<td>3000 300</td>
<td>278 0</td>
</tr>
<tr>
<td>offp:</td>
<td>100.0 10.0</td>
<td>10.0 12.0</td>
<td>50.0 100.0</td>
<td>300.0 20.0</td>
<td>2000 200</td>
<td>0 0</td>
</tr>
</tbody>
</table>

Value of a Rider: 50.0 (cents)

Use < - > to move cursor
Cntl-C to accept data

Esc to MAIN menu
Cntl-P to print screen
## FIGURE 3
THE OPERATING SCREEN

<table>
<thead>
<tr>
<th>DATA</th>
<th>Operating</th>
<th>File: GEORGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corridor Length (mi)</td>
<td>8.0</td>
<td>Maximum Passenger Load/Bus (s)</td>
</tr>
<tr>
<td>Corridor Width (mi)</td>
<td>6.0</td>
<td></td>
</tr>
<tr>
<td>No. of Expressways</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>CBD Length (mi)</td>
<td>0.5</td>
<td>Local Express</td>
</tr>
<tr>
<td>Analysis days/year</td>
<td>250.0</td>
<td>peak: 40.00</td>
</tr>
<tr>
<td>Total/Serv. Bus-Hrs.</td>
<td>0.80</td>
<td>offpeak: 40.00</td>
</tr>
</tbody>
</table>

### Bus Operating Cost
- Peak-only: 36.00 (.6c1)
- Base period: 33.00 (.6c3)

### Length of Period (hrs)
- For peak analysis: 4.00 (T1/60)
- For offp analysis: 7.00 (T3/60)

### Average Segment Speed (mph)
- Local: 12.00
- Express (w/stops): 13.00
- Express (w/o stops): 33.00
  - Offpeak: 33.00

Use <-, -> to move cursor
Cntl-C to accept data
Esc to MAIN menu
Cntl-P to print screen
with one decimal point. An example of the Existing screen is shown in figure 4. These data are used as the basis for predicting changes in service levels of alternative operating plans, which in turn determine changes in ridership and revenue.

The Market Screen. This screen keeps estimates of the following data: walk speed, maximum distance users are willing to walk, CBD parking costs, maximum policy headway of the transit system, the ratio of wait time to headway, and the sensitivities of ridership to fare, ride, walk, and wait time. All data but the sensitivities are real with one decimal place; the sensitivities can have up to six decimal places. The screen is illustrated in figure 5. The sensitivities are determined by choosing from a set of curves in the manual. Default values are also suggested. Although we intended to display these curves on the screen, the limitations of the Apple II and the screen-handling procedures made this too cumbersome to implement.

3.3.4 Related Functions.

FRACAS thus operates with six interactive screens. Each screen displays a related body of data or choices which can be entered, modified or verified by the user. The computer model verifies all data for completeness and correctness before computing any results. Although most errors are detected when the analyst accepts the screen with Cntl-C, some error-checking requires data across several accepted screens — this checking is done before FRACAS processes the input data. Any errors or omissions detected will cause the program to return to the affected screen and position the cursor on the problem. A message will be displayed at the bottom of the screen.
## FIGURE 4
THE EXISTING SCREEN

<table>
<thead>
<tr>
<th>DATA</th>
<th>Existing Service</th>
<th>File: GEORGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Routes</td>
<td>[(r)] Local (6.0) Express (100.0) Total Ridership (5430) Express (500)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(r) peak: 6.0 (\text{offpeak: 100.0}) (d) peak: 5430 (\text{offpeak: 400})</td>
<td></td>
</tr>
<tr>
<td>Route Length (mi)</td>
<td>[(u)] peak: 7.0 (\text{offpeak: 20.0}) (100n) peak: 80.0 (\text{offpeak: 50.0})</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(u) peak: 7.0 (\text{offpeak: 20.0}) (100n) peak: 80.0 (\text{offpeak: 50.0})</td>
<td></td>
</tr>
<tr>
<td>Fare (cents)</td>
<td>[(e)] peak: 70.0 (\text{offpeak: 100.0}) (100m4) peak: 78.0 (\text{offpeak: 60.0})</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(e) peak: 70.0 (\text{offpeak: 100.0}) (100m4) peak: 78.0 (\text{offpeak: 60.0})</td>
<td></td>
</tr>
<tr>
<td>No. of Bus Trips</td>
<td>[(t)] peak: 90.0 (\text{offpeak: 130.0}) (100z) peak: 20.0 (\text{offpeak: 30.0})</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(t) peak: 90.0 (\text{offpeak: 130.0}) (100z) peak: 20.0 (\text{offpeak: 30.0})</td>
<td></td>
</tr>
</tbody>
</table>

Use \(<, >\) to move cursor

Ctrl C to accept data

Esc to MAIN menu

Ctrl-P to print screen
### FIGURE 5
**THE MARKET SCREEN**

<table>
<thead>
<tr>
<th>DATA</th>
<th>Market</th>
<th>File: GEORGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. Walk Speed (mph): 3.0 (60j)</td>
<td>SENSITIVITIES</td>
<td>select from graphs in the manual</td>
</tr>
<tr>
<td>Max. Walk Distance (mi): 0.5 (H)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg. CBD Parking Cost (cents/trip):</td>
<td></td>
<td>Local Express</td>
</tr>
<tr>
<td>peak: 150</td>
<td>Fare</td>
<td>pk:.002000 .002000</td>
</tr>
<tr>
<td>offp: 50</td>
<td>(a4)</td>
<td>offpk:.002000 .002000</td>
</tr>
<tr>
<td>Max. Headway Policy (Z) (in min):</td>
<td>Local Express</td>
<td>Running Time</td>
</tr>
<tr>
<td>peak: 60.0 60.0</td>
<td>pk:.003000 .003000</td>
<td></td>
</tr>
<tr>
<td>offp: 60.0 60.0</td>
<td>(a3)</td>
<td>offpk:.003000 .003000</td>
</tr>
<tr>
<td>Wait-to-Headway Ratio (k):</td>
<td></td>
<td></td>
</tr>
<tr>
<td>peak: 0.40 0.40</td>
<td>Walk Time</td>
<td>pk:.010000 .010000</td>
</tr>
<tr>
<td>offp: 0.40 0.40</td>
<td>(a8)</td>
<td>offpk:.010000 .010000</td>
</tr>
<tr>
<td></td>
<td>Wait Time</td>
<td>pk:.010000 .010000</td>
</tr>
<tr>
<td></td>
<td>(a2)</td>
<td>offpk:.010000 .010000</td>
</tr>
</tbody>
</table>

Use `<`, `>` to move cursor
Cntl C to accept data
Esc to MAIN menu
Cntl-P to print screen
There is also a "help" facility in FRACAS. This facility will display a full screen of information for any data item on the five screens containing input data. The facility is invoked by pressing Cntl-Q. The help screen will describe the name, type, decimal places, and range of the data value, plus give a prose description of the variable. The user returns to the screen that he or she was working on by pressing Cntl-C. Because of limitations with the screen-handling procedures, the lines on the data screens will not return after a help file has been examined.
SECTION FOUR
SYSTEM DEVELOPMENT

4.1 Introduction

This chapter describes the internal structure and coding of FRACAS. The optimization procedures used, reported in Kocur (1982), were translated directly into UCSD Pascal. The input/output routines were based on a software product known as the UMTA Screenhander. The total length of the program (including the Screenhander) is nearly 6,000 lines. This section explains the process necessary to put FRACAS on a microcomputer.

4.2 Hardware and Software Selection

The first decision to be made in implementing the software specification was to choose a microcomputer for FRACAS. Knowing that the program would be relatively large, there was temptation to use a powerful machine. However, there were other considerations.

Although the power of a microcomputer does affect the size of the program that can be run, the ability to break up large programs into smaller sections that can be sequentially executed usually implies that a more powerful machine will simply run a particular application faster. Since the speed of an application can also be somewhat increased through additional efforts in programming, the typical trade-off is between the cost of the machine and the effort necessary to develop an acceptable system. For a given application, the less costly hardware
requires more expenditure of effort in software development.

The costs to the end user can thus be minimized by implementing the system on a small (and inexpensive) machine, such as the Apple II. Second, and more important, the Apple II is likely to be a machine that is often available in transit agencies. For these reasons, the Apple II was chosen as the hardware for FRACAS. In retrospect, a larger machine such as the Apple /// might have been a more appropriate choice. However, there appeared to be no machine that had the power of the Apple /// with the popularity and acceptance of the Apple II.

The second decision to be made concerned the language to use for program development. The two languages that are currently popular with the Apple II are Pascal and Basic. The Pascal is a version of UCSD (University of California, San Diego) Pascal, which emphasises the portability of compiled code between different types of computers. Thus, a system developed in Pascal would theoretically be easy to move onto other hardware. This was a key consideration in favor of using Pascal. In addition, Pascal is a compiled language, and although this increases development time, the application runs faster. Also, Pascal has many more features than BASIC, including real subroutine capabilities, and it supports true structured programming. Last, UMTA suggests Pascal as an appropriate language for microcomputers, and the UMTA Screenhandler will operate only with software written in Pascal. We chose Apple Pascal, version 1.1, for FRACAS.

The number of columns available on the screen will obviously affect the amount of information that can be put on one screen and consequently the ease of use of
the system. Since both the Pascal system and the UMTA Screenhandler support the
addition of hardware to the Apple that expand the number of displayed columns
from 40 to 80, a Videx Videoterm was also specified as part of the hardware
package that runs FRACAS. The Videx was chosen because it is one of the most
common 80 column boards available and would be most likely to be part of
existing equipment belonging to transit operators. Since FRACAS makes use of
the simple line graphics available to the Videx board, the Fracas system is not
compatible with other 80 column hardware.

4.3 Implementation of User Interface

4.3.1 Use of the Screenhandler.

The UMTA Screenhandler is both a set of utilities and a library of procedures
for the easy implementation of interactive screens and menus. This software is
available to developers of transportation software for microcomputers for the
federal government. All of the interactive screens except the RBPORT screens
were designed with Screenhandler utilities, and operate by calling the
Screenhandler procedures.

The use of the Screenhandler in software development to prepare interactive
screens is also user-oriented. An interactive screen is produced in two parts:
the background text that is static during screen operation, and the data entry
locations. The background text is developed using the text editor provided with
the Pascal system, and stored on disk. The data entry locations are developed
with a menu driven utility program that prompts for the name, data type, number
of digits and other specifics of each data entry location. This "data field
definition" information is also stored on disk.

A second utility program within the Screenhandler, called the binding program, reads both the background text file and the data field definition file, displays the background text and allows the user to point out the data entry locations with the cursor, and binds this information into a format file. This format file contains all the information about the interactive screen needed by the executing program and must reside on disk while the program is executing. The Screenhandler will optionally generate a help file. When this file is made present to the executing program, the user can easily be prompted with help information if requested. FRACAS makes use of this help file capability.

4.3.2 Data Organization.

The UNTA Screenhandler is designed so that the current values of screens not currently being displayed are stored as part of the format files residing on disk for these screens. In FRACAS, all of this information regarding present screen values is referred to as the "workspace." The user perceives the workspace as being the data which are currently being edited.

Although screen data are stored in the Screenhandler format files on disk, the data are reorganized when brought into machine memory into an array of integers and three arrays of reals. The three arrays of real differ by the size of their second dimension. Thus, one matrix is used for scalars, while the second and third are used for data that are naturally grouped into sets of four or five data items. By organizing the data into matrices, the handling of the data is simplified. For example, Pascal can store an entire matrix with one command,
thus greatly reducing the complexity of disk access procedures.

Since the format files contain background text, maximum and minimum values for each data entry location, and much other information, they are inefficient means for the storage of data on the screen. For this reason, when the OUTPUT option or the STORAGE page are selected, each menu format file is called once, and the four FRACAS data matrices are updated. It is these matrices that are stored to disk for use by the OUTPUT program, or stored by the STORAGE page as record of the menus status for a particular run.

The typical flow of data starts with the information contained by a completed set of interactive screens residing on disk in the format files. If the STORAGE page is selected, all five format files are read, and the information is converted into the matrix form. When files are stored, replaced, or loaded, these matrices are combined into one record structure — thus each workspace that is stored puts only one file onto disk. This avoids the unpredictable performance that might occur if a disk error occurred after storing one of several files necessary to record all the menus. It also saves on storage space.

Likewise, when OUTPUT is selected from the MAIN menu, the workspace information is collected from all the format files and put into the matrix format. This information is stored on the first disk in four separate files. These files contain zeroed spaces for all the outputs generated in the OUTPUT program. The OUTPUT program retrieves this data, processes it, and rewrites the new information over the existing disk files. The REPORT program is called, which reads these files and displays them to the screen. The REPORT program then
calls FRACAS, and the user is free to modify any data with the interactive screens, and then repeat the process.

4.4 FRACAS System Structure

The FRACAS system is divided into four programs responsible for the input menus, calculations and output reporting. When FRACAS passes control from program to another program, the Apple’s internal memory is reset. Thus, any data needed must be stored on disk.

The first program is only run when the system is turned on. This program displays the title and calls the input program. This allows the title to be on the screen while all the initial calculations are being done. The inputs program handles the interactive input screens, runs the STORAGE page, and checks the input data and writes it on disk before chaining to the output program (OUTPUT). The output program reads the input data from disk, calculates the output variables, and writes the output data to disk before chaining to the reporting program (REPORT). This program reads the output data from disk and displays it on the screen. The report program chains back to the input program. This overall structure is displayed in figure 6.

FRACAS is the largest program (it includes the Screenhandler procedures) — OUTPUT is virtually as large. TITLE is quite small, and REPORT is about half the size of FRACAS. TITLE and FRACAS are on the first disk along with the Screenhandler format and help files, while OUTPUT and REPORT are on the second disk.
FIGURE 6
PROGRAM TASKS

TITLE

FRACAS
Input Screens
Storage Management
Input Error Checking
Writes Input to Disk
Chains to OUTPUT

OUTPUT
Reads Input From Disk
Analyzes Input, Produces Output
Writes Output to Disk
Chains to REPORT

REPORT
Reads Output from Disk
Writes Report to Screen
Can Print Report
Chains to FRACAS
4.5 Programming Considerations

The memory space required for the compiled code for the FRACAS system is about 99k bytes, including space for the data heap and the necessary library routines. The available space in the Apple II for a program, after the Pascal system has been loaded, is about 39k bytes. The code must therefore be divided into sections that can be loaded in memory.

4.5.1 Chaining.

The first mechanism for dividing up the code file is to design the system as several separate programs that are called sequentially. Apple Pascal supports a Chaintuff unit, which makes the procedure "Setchain ( <filename> )" available for use in the user's program. At the normal termination of the calling program, the program "filename" will automatically begin, with no disturbance of the terminal screen; hence, the user of the program is unaware that the chaining occurred.

The code of the FRACAS system is separated into four codefiles. The first codefile, FRACAS:TITLE is executed only when the system is turned on, and displays the title screen and moves on to the second codefile. The second codefile, FRACAS:FRACAS, handles the input menus, STORAGE page, and checks the input data before calling the next codefile, FRACAS2:OUTPUT, which performs the actual numerical calculations. OUTPUT calls FRACAS2:REPORT, which displays any of four output pages to the screen, and can print these results. Leaving the REPORT program returns the user to FRACAS.
The choice of how to divide the code for chaining is not difficult, but a poor decision can impair the responsiveness of the system. Since each move to a new codefile takes about five seconds, it is imperative that this be done at natural rest points in the execution of the system. Since the optimization code already provides a processing delay, choosing the chaining operations to come before and after the execution of this code presents the least objectionable delays to the user. This division also allowed the system to be programmed in component programs that were relatively independent, thus simplifying the debugging procedure.

4.5.2 Segmentation.

Although the chaining of the FRACAS system does reduce the system into more manageable sections, the resulting programs are 2K, 4K, 42K, and 11K large, so that two of the remaining pieces are still too large to fit into the Apple’s memory. Luckily, Pascal also supports segmentation, which allows the programmer to specify that certain procedures in the source listing reside on disk until they are called. By selecting two procedures that never need to be in memory at the same time and specifying them as segment procedures, the memory needed to execute the codefile decreases. In Apple Pascal, segment procedures are specified by placing the word "Segment" in front of the procedure declaration. Segment procedures must be the first procedures in the source to generate code.

The choice of which procedures to declare as segment procedures is not as easy as that of choosing program chain points. Since disk access is necessary each time a segment procedure is called, it is unadvisable to declare frequently called procedures as segments. The larger the procedures that are declared as
segments, the less segments are necessary to bring a program down to an executable size. Although this is somewhat offset by the fact that the larger a procedure is, the longer it takes to get from disk, the overall objective is usually to minimize the number of disk accesses required.

Two segments were chosen for FRACAS. The first included all the code necessary for the STORAGB page. Since the STORAGB page already must get five format files when being called and store five format files when returning to the MAIN menu, the access time to get the procedures from disk will be less noticeable. Likewise, the second segment contains the code that checks for all needed data before chaining to the OUTPUT program, a sizable piece of code that only executes directly before the relatively long wait for the report pages.

The OUTPUT program can be thought of as several procedures that do preprocessing, a set of procedures that do the optimization, and several procedures that do postprocessing. The nine optimization procedures represent different "cases"; only one is executed in a particular run. By segmenting the optimization procedures, a considerable amount of code is kept on disk—usually at the expense of only one disk access. When the objective of maximizing ridership within deficit constraint is requested, however, the "case" procedures are iterated; in this case, the program is delayed a few seconds for each iteration. This delay can be observed by watching the iteration counter messages displayed during program execution, allaying fears that the program is running too long.

Unfortunately, the segmentation of the optimization procedures does not alone reduce the size of the resident code enough to execute the program. For this
reason, the preprocessing procedures are also segmented.

4.5.3 Use of Units.

A unit is a group of procedures that are precompiled and kept in a library for use in other programs. Thus procedures that are frequently used in many programs can be compiled once and "linked" to new programs. Units are a unique feature of UCSD Pascal.

Unit is written in two sections. The first section contains all the declarations and comments, the former needed so that the computer can set aside the proper amount of space for data; the second section contains the actual code. The first section is copied directly into the program using the unit (called the "host" program) during compilation, and the code is linked to the program after compilation. Only the declarations and comments are available in a printout of host program — the actual code cannot be displayed.

The UMTA screen handler is a unit. The declarations and comments can be seen at the beginning of the listing for FRACAS. By using the unit, procedures such as "Loadform" and "Showform" can be used without specifically writing the code for them. The effect is that the number of commands "native" to Pascal increases.

Another unit, called DISKSTUFF, was written specifically for FRACAS. This unit extends the Pascal language to contain commands that send data files to disk, put a catalog of datasets on the screen (used on the STORAGE page), draw lines on the screen, and turn the cursor on and off. The declarations for this unit are also seen at the beginning of the FRACAS listing.
Two other units are used by the programs in the FRACAS system. These are CHAINSTUFF and TRANSCEND. CHAINSTUFF adds commands that allow a program to chain to another program (see "Chaining" above), while TRANSCEND adds commands that handle the exponential, logarithmic, square and square root functions in FRACAS. Unlike the units described above, CHAINSTUFF and TRANSCEND are provided with the Apple Pascal system and do not have their codefiles linked to the user's compiled program. These units are in a library called 'system.library', and reside on disk much like segmented procedures. The Pascal system loads these units at run time. This requires that "system.library" be on the program disk during execution.

For this reason, there is a file "system.library" on the disk "FRACAS". However, since space is at a premium on this disk, the library is NOT the same library as the one provided with the Apple Pascal system. It contains only PASCALIO, TRANSCEND, and CHAINSTUFF. All units unneeded by FRACAS have been purged. It is important not to attempt to use this library as you would typically use the original "system.library".
The goal of the development of any system such as FRACAS is to improve the decisions made in the application environment. For this reason, the discussion of the development of a transit model should extend beyond the programming of the model. This section presents a discussion of a few issues that will have an effect on the usefulness of FRACAS in the field.

5.1 User testing

After the model coding was completed, the model was demonstrated to several people to solicit their opinions and reactions. At the present time, the model is still in this stage of its development.

Typically, a potential user was tested "hands-on". After a brief introduction to the system, the user attempted to use his or her intuition to enter data and experiment with the system, attempting to replicate a transit route or system that they were familiar with. This testing helps us with the refinement of the program in three ways:

1. The familiarity of the user with the variables specified in the system can be observed. This allows for the possible refinement of variable specifications for refined releases of the system. For example, if a large number of users have trouble specifying the ratio of total bus hours to in service bus hours, it may be prudent to have the user specify layover and deadheading times explicitly and have the model,
using the appropriate assumptions, determine this value internally.

2. The user's understanding of the system in general can be observed.
   Things such as the purpose of each of the screens and the flow through
   the screens should be intuitive as best, or at least easily learned.
   Our observations so far reinforce the thought that the purpose and
   overall functioning of the model are easy to understand.

3. The "user-friendliness" of the system can be assessed. One original
   goal for FRACAS was to make the system straightforward enough to operate
   so that non-programmers would not be discouraged with using the system.
   Some minor items have already been noted that detract from the otherwise
   good friendliness of the system: 1) because of the type ahead buffer, a
   user can essentially lose control of the program because of inadvertent
   pressing of the ESCAPE key. 2) the use of the ESCAPE key to select an
   option off of the MAIN menu was found to be awkward by several users.
   The system makes repeated use of Ctrl-C for selecting and accepting, and
   the general consensus is that the MAIN menu should conform. A secondary
   problem with using ESCAPE for selecting is that people seem to get used
   to the keystroke and continue to use it. Users have "erased" a page of
   data that they just entered by pressing ESCAPE after entering data. Our
   lesson is to use ESCAPE only for "escaping" to the MAIN menu without
   affecting the workspace.

5.2 The need for FRACAS

One of the original motivations for the development of FRACAS was that transit
may be facing a time of increasing fiscal tightness, and along with this
tightness will arise needs for new methods in planning. Among other things,
Michael Meyer and P. Brendon Hemily conclude that this new "fiscal austerity"
will bring with it a greater need for strategic planning and a long term
perspective on the change that is occurring [Meyer and Hemily, 1982]. They also
suggest that another coming aspect of transit planning will be the generation of
a fuller range of alternatives. They consider the process of generating these
alternatives as a "learning process for management that widens perspective and
refines intuition."

Although these items mentioned are only a piece of the total picture for transit
planning of the future, it is a piece that can be addressed by FRACAS. Because
of its interactive design, FRACAS can be a part of the process of creating,
refining, and analyzing new route configurations and fare structures. The
obstacles that lie ahead for FRACAS are not as much that there is no need for
such a system, but that the system may not be accepted to do what it is designed
to do. We will now look at some of these potential problems with
implementation.

5.3 Perceived Implementation Problems

The list of implementation problems that will potentially plague a particular
project is virtually infinite in length. However, by examining the environment
that currently exists in transit agencies and discussing the experiences of
others who have tried to implement transit planning methodologies, some
particular implementation problems have been brought out as significant. These
are presented here to make the reader more aware of the potential problems and to suggest approaches to minimizing the effects of some. The problems perceived at this point in time are:

1. FRACAS presents a radical departure from the traditional methods of determining future services and establishing proper fares. The current planners and managers who rely on years of experience and intuition, plus a detailed knowledge of the specific system they work for, may reject or even resent the inclusion of input from a model that was developed outside of the boundaries of the transit property. Thus, it may be difficult to get the model accepted in the office or to have the model receive respect or even attention once it is installed.

In considering this implementation problem, we must admit that this problem will exist to some degree. Resistance to change, though, is not new, and is certainly not a good reason to give up an attempt to bring about change. In fact, the potential "crisis" atmosphere that may occur in some transit agencies as federal funding becomes more scarce may be a catalyst for the integration of such new technologies. There will be a renewed interest in the increasing of system efficiency and the ability to quickly look at a broader base of alternatives.

2. The current process for decision of fare levels and extent of service is a political one. FRACAS is attempting to change this to inject technical content and this will be hard to do.

Several comments are in order. First, we are not attempting to
"convert" the decision process to a technical one, since we are not recommending that the outputs of FRACAS be accepted "as is." Rather, we are suggesting that a new technical tool be added to the existing process for the new insights it may bring. Second, it is not clear that all decisions currently being made by the political process must be made by the political process — there may be issues from which the political process would like to retreat and leave to a more technical level.

3. The underlying assumptions can be and will be criticized, resulting in a lack of confidence in the model.

The model does not have an exact interpretation of the physical aspects of the transit system. However, the driving limitation here is the data. With only route level and operating data being used at the strategic level of planning, many detailed variations are not addressed. Although the methodology of the model can treat such detail, it makes the model too data-intensive and cumbersome to use. It is also important to realize that the actual assumptions that have been made have been designed to have as little impact on the final results as possible. The planner who rejects the model on its assumptions is perhaps rejecting the concept of gathering intuition on a system design from the amount of data required by FRACAS. There are reasons to doubt the accuracy of a particular run in this case, but given the continual problems planners have with lack of data, we argue that FRACAS can substantially improve the understanding of the trade-offs facing a property.
4. There is little present tendency to do strategic planning. For this reason, FRACAS will not be seen as useful in the current planner's or manager's workday -- especially when considering the busy workload already burdening these people. For this reason, even if the model is accepted by transit operators, it may fall into disuse.

This is a genuine problem. The factors that would help to surmount these problems include: 1) the changing fiscal environment of transit operations may create an environment of open-mindedness as managers look for new ways to be efficient; 2) the interactivity and ease of use will allow users to become exposed to FRACAS in a matter of minutes, so that curiosity (even after-hours) may become a significant part of the integration process; and 3) it may show sufficient usefulness in crisis situations to convince a property to implement it on a continuing basis.

5. People lack experience with microcomputers and are consequently apprehensive about the use of such equipment.

This is now true, but becoming less so each year. The user-friendliness of this system is good, so the real problem may be getting people to try it.

5.4 Conclusions

There are a large number of potentially severe implementation problems for
FRACAS. In general, these focus on the interpretation of FRACAS as an attempt to handle the task of strategic decision making internally, instead of viewing FRACAS as a technical aid in the development of intuition of the planners and managers making the decisions. For this reason, some implementation tactics are suggested at this point:

- It should be emphasized that the real purpose of FRACAS is to increase the efficiency of the planners and managers using the system. Rather than marketing the system as an optimization, it would be better to market it as a technical aid or a decision support system, emphasizing the interactive, "what-if" qualities of the model.

- The paucity of the required data should be emphasized, pointing out that this makes the model more useful in many cases. The assumptions that are made are designed to reduce the amount of data needed, and are relatively robust with respect to the output variables. In addition, the small amount of data should increase the transparency of the results, leading to better intuition about the trade-offs that exist.

- The environment of "fiscal austerity" may be an inadvertent help to implementation. The model can be billed as an effective tool for vastly speeding up the process of finding acceptable alternatives for detailed "political" analysis as well as providing some measures of performance from a "technical" analysis. Ultimately, the FRACAS will be evaluated on the experiences of managers and planners who use it, but effective marketing of this tool as an efficient way to assist the existing planning process should ensure its best chances of impacting transit strategic planning.
APPENDIX A
FRACAS USER'S GUIDE
VERSION 2.0
MAY, 1983

A.1 Introduction

A.1.1 Program Purpose

The program FRACAS (Fare and Route Analysis - Computer Aided System) is
designed to interactively help transit managers and planning staff with the
task of establishing fare and service policy. Since the output procedure
typically takes ninety seconds or less, the model can be run repetitively
to develop an understanding of the fundamental choices affecting the
performance of a particular transit system.

For a given transit system corridor or route, FRACAS specifies a
combination of service and fare that best achieves system objectives. The
service is defined by the number of routes in the analysis, their average
length, the average headway and the load per bus. Express and local
service, and peak and offpeak time periods can be treated jointly. FRACAS
computes the service levels, ridership, revenues and costs of options, and
provides statistics on bus-miles, bus-hours, and passenger-miles, for
example.

Specification of the service area is quite general: a corridor within the
system or even a particular route can be specified. In addition, the objective the model works towards can be modified, as well as the number and choice of variables that the model is given control over. Thus, the model can specify the optimum fare for a given service for a particular route, or perhaps the optimum headway for a given fare. In the extreme, all variables can be specified. In this case, FRACAS simply operates as an evaluation tool, estimating ridership, revenue, cost and service impacts. In all cases, FRACAS will estimate a full set of cost and revenue statistics for the service specified.

A.1.2 Required Equipment.

The system specified for use with FRACAS includes an Apple II (the Apple II Plus with autostart is preferred, but not required), a monitor capable of displaying 80 columns (a T.V. set will not usually be sufficient), two DISK II datadisks and interface card, a language card (or equivalent slot 0 Ram expansion card with at least 16k bytes storage), and a VIDEX VIDEOTERM brand 80 column video interface mounted in slot 3 (other 80 column boards will give unpredictable results). And a standard size printer is highly recommended. Since the software is written with the Apple Pascal system, the user experienced with the Pascal system may find it useful to have the system available, although the Pascal system is not required to run FRACAS.

A.1.3 Getting Started.

Before turning on the machine, insert the diskette labeled "FRACAS" (or hopefully a copy of FRACAS; if you must use the original, take off the
write protect tabs. Instructions for copying diskettes are below) in the first disk drive, with the label facing upward and on the edge of the diskette closest to you. Likewise, put the second diskette into the second drive. These diskettes will typically stay in place throughout the worksession.

If your system is an Apple II plus (so designated by the green letters "plus" on the top of the machine), just turn the power on. The power switch for all Apple II's is on the left rear corner of the machine. If your Apple II is not a "plus", then follow the instructions in your manual for starting up the system. You may try typing "PR#6". If the disk drive starts up, you've succeeded.

At this point, the title page will appear, followed by the MAIN menu. Some precautions will provide the most satisfactory performance:

1. Although the system is designed to be interactive and respond as quickly as possible, it does take about five seconds to move from menu to menu, about thirty seconds to move to and from the STORAGE page, and about ninety seconds to get to the results. Please be patient. Most users, knowing what is coming on the next screen, use this time productively preparing for coming questions.

2. The software, by virtue of the fact that it is written in Pascal, has a type-ahead buffer, and will store all keystrokes that it is unable to act on immediately. Thus, if you repeatedly hit the ESCAPE Key because nothing seems to be happening, the program will spend the next minute
paging through all the menus. There is no way to regain control of this situation other than letting the system come to a rest, and continuing from there. The inexperienced user will typically make this mistake only once. The experienced user, in fact, will learn to take advantage of this feature.

3. There are several keystrokes that are always available and should be remembered. ESCAPE always returns the user to the MAIN menu. When on the MAIN menu, ESCAPE will select an option. Cnt1-C always accepts the screen: on the MAIN menu, Cnt1-C still selects; on the data screens, Cnt1-C will put the data on the screen into the workspace; on the STORAGE page, Cnt1-C selects an option; and on the report pages it does nothing. Cnt1-P will always print the screen, although printing the catalog of datasets is done by selecting the separate option for that purpose on the STORAGE menu. Cnt1-Q will request help at any time, although no help will be provided on the MAIN menu, the STORAGE page, or the report pages. The lines on the data pages will not return after returning from a help file — this is normal operation. In addition, these lines will not appear in printouts of these screens.

A.2 Program Structure

The FRACAS program structure is shown in the figure on the following page. From the MAIN menu, one can select the OBJECTIVE menu, any of the data menus (CONSTRAINTS, AREA, EXISTING, MARKET), the STORAGE page or the OUTPUT routine. Each will return to the MAIN menu upon termination. QUIT ends the program.
A.2.1 The Interactive Screens.

The OBJECTIVE menu and the four data menus are for the entry of data about the transit system being analyzed. They are designed to provide a quick way to specify a complete system from scratch, but also allow a user to return to the specification and quickly update just one or a few pieces of information within it. It is also possible to print out the information on any menu.

The OUTPUT selection invokes the optimization program on the second diskette. The screen will clear, and the user is provided with warning messages about the processing of his or her data. If there is a piece of needed information missing from one of the menus, the program will temporarily return to that menu to get the needed information. Upon the completion of the optimization procedure, the results will be displayed on the monitor. The user can move back and forth through the four pages of output, and then return to the MAIN menu, either by printing a one page report of that run or using ESCAPE.

Upon returning to the MAIN menu, the user can move into the STORAGE page to save the data used to create the output. This is the only way of preserving a record of the data used to create a particular output, short of printing out each of the individual menus. The system can store over seventy runs on the original diskettes; a copy of the second diskette can be used for more space. The STORAGE page can be used at any time to store, retrieve or update the files on disk. The page will also provide the user.
with the current catalog of datasets, which can be printed.

A.2.2 The Workspace.

You will notice on the upper right corner of the MAIN menu that a "File" is specified. This is the name of the current workspace. The workspace contains all the data on all of the menus. The information contained in the workspace is always maintained on disk. When the screen is exited in the normal fasion (using Cntl-C), the workspace is updated. When the user exits the screen with the ESCAPE key, the new information is lost, and the workspace continues to contain the original workspace information. In this way, the system can be turned off, and when "re-booted" contain the workspace from the previous session.

This feature can also be helpful during an unexpected power loss. In this case, only new data being input to a screen would be lost. The old version of that screen and all other screens would still be intact. In this case the name of the workspace may revert to the name in the previous session, since it is the "Quit" procedure that stores the name of the current workspace to the disk. If this happens, the name of the workspace can always be changed by simply moving the cursor to the filename and typing the new name. The new name will be recorded when you accept (Cntl-C) the screen.

A.3 Running FRACAS

Running FRACAS is straightforward: the user pages through the interactive
screens and inputs the data needed. When finished, the OUTPUT selection is made and the results are displayed. Any information available on any screen can be printed at any time. Any data in the workspace can be stored under a unique name at any time.

A.3.1 Data Entry.

Each location to which the cursor moves on the OBJECTIVE menu and the four data menus is a data entry location. For a typical run, the system may need about 50 pieces of information entered into the data entry locations. When "pivoting" off of a previous run, the user may only need to modify a few values.

For each of the data entry locations, there is a specification of the data that is allowed to be entered there. This specification lists the data type, the size of the field and the number of decimal places for real numbers, whether the entry is required to accept a screen, and the range of numerical data. This specification is part of the help page that can be seen if you type Cntl-Q.

The arrow keys on the right side of the keyboard (\(\text{\textasciitilde} \& \langle \text{\textasciitilde}\rangle\)) are used for moving the cursor among the data entry locations on a given menu. Data is entered by keying the desired information in with the keyboard. After all data on a particular screen has been entered, the data is accepted by typing Cntl-C. If at any time the user wishes to leave the screen without accepting the new data into the workspace, the ESCAPE key will cause the current menu to be exited leaving the previous contents of the screen.
intact.

A.3.2 The STORAGE Page

The STORAGE page is used to send a copy of the OBJECTIVE menu and the four data menus to disk. Each file sent to disk is given a name; a list of these names, called the catalog, appears on the storage page when the screen first appears. This catalog is updated in "real-time", that is, it is continually updated with the files that are on the diskette.

The Pascal volume number of the device with which to send and retrieve data can be specified by selecting the "catalog" option. This option will prompt you for the volume to be catalogued, update the on-screen catalog, and subsequently send and retrieve all data to that device. The permissible numbers are 4, 5, 7, 8, 9, 10, 11, and 12. See page 26 of the Apple Pascal Operating System Reference Manual for a more detailed description of volumes in Pascal [see references]. The default volume is 5, which corresponds to the second disk drive in the typical two drive set up. The specification of another unit is not useful unless additional floppy drives or a hard disk have been added to the system.

Operation of the STORAGE page is simple. With the cursor on the select position, enter an option. The cursor will automatically advance to the "filename" prompt, in order to enter the name of the file you wish to store, replace, load or delete. The filename specification is ignored for options 5 & 6. Note that the system responds to a request to delete a file with a question requiring that you verify that you wish to delete the
specified file.

A.3.3 Output.

Selecting OUTPUT from the MAIN menu will cause about ninety seconds of activity from the computer, ending with the first page of the Management Report being displayed. Using the arrow keys, the user can look through all four pages of output, returning to pages that have already been viewed if desired. The only keys on the keyboard that are active at this time are the two arrows, the ESCAPE key which returns the user to the MAIN menu, and Cntl-P, which prints one page of output and then returns to the MAIN menu.

The selection of OUTPUT from the MAIN menu actually brings about a chain of activities. In the order of execution, these activities are:

1. Each of the menus is called from disk (but not displayed) and the data is stored in a compressed format in memory.

2. This data is examined to make sure that all needed information is available. If additional information is necessary, the program will temporarily move to the menu concerned, flag the data entry position that requires data, and prompt the user to enter the data. If this is followed by Cntl-C, the OUTPUT process continues; if the ESCAPE key is pressed, the OUTPUT procedure is aborted and the user is returned to the MAIN menu.
3. If all needed data is present, the compressed data files are written to the system files on disk "FRACAS".

4. The system clears memory and "boots up" the program OUTPUT from the second diskette.

5. The data in the system files is retrieved.

6. The optimization code is run and the output computed. Warning messages, if any, are sent to the screen -- these cannot be printed.

7. The output of this optimization is written to the system files on the disk "FRACAS".

8. The memory is again cleared and the REPORT program is called from the second diskette.

9. Data is retrieved from the system files. The REPORT program displays the data, then chains back to FRACAS when ESCAPE is pressed, and prints if Cntl-P is pressed.

A.4 The Diskettes

There are two diskettes on which the FRACAS system resides. The first disk, "FRACAS" contains the master program FRACAS, one format and one help file for each of the interactive screens, a small program that puts up the
title screen, and four system files used to hold data while control is passed from one program to the other. The second disk, "FRACAS2" holds the programs OUTPUT and REPORT, plus an additional program called DUMPPRT. The remainder of the "FRACAS2" diskette is used for the storage of data by the STORAGE page.

The system is designed to operate with "FRACAS" always in drive 1 and "FRACAS2" always in drive2. Since the FRACAS system makes extensive use of these diskettes, it is strongly recommended that you use the original diskettes only as backups, and that you begin your first session by backing up these diskettes and putting the originals in a safe place. Note that the originals have been shipped with write-protect tabs and cannot be used (only copied) with these tabs in place.

There is a two step process involving both the initiation of a blank diskette into Pascal diskette and the transferring of material onto that diskette. The Apple II DOS (Disk Operating System) will not copy these disks. All files needed are part of the Pascal system. The process is briefly described here.

What's required:

A blank (or useless) single sided, soft sectored, 5 1/4" diskette. Two are needed to copy the FRACAS system.

A Pascal diskette with SYSTEM.FILER on it and a diskette with the utility FORMATTER on it.

The steps are:

1. Quit FRACAS. Place the diskette with
FORMATTER on it in the top disk drive.

2. Press the "X" key. The system will respond, "execute which file?".

3. Answer,"#4:FORMATTER" <return>.

4. The system will eventually respond, "format disk in which drive?".

5. Place a blank disk in the second drive. Respond to the question with "#5" <return>.

6. Likewise, format a second disk, if desired. End the FORMATTER program by pressing return.

7. Put the diskette containing SYSTEM,FILER into the first drive. Press "F", and then "T".

8. The system will respond by asking "transfer which file?". Place the source disk in the first drive and the blank disk in the second drive. Respond to the question with "#4,#5" <return>. Respond "Y" to questions asking if you wish to copy 280 blocks and destroy "BLANK". The system will copy the disk.

9. Repeat steps #7 and #8 to copy additional disks. When finished, put the FRACAS disks in their proper drives and type "Q". Then type "X" followed by "title".

A.5 An Example Run

In this section, an example of the FRACAS model is run. This example is also described as example 1 in A Unified Approach to Performance and Fare Policies for Urban Transit Systems, which documents the manual analysis version of FRACAS.

This example is described step-by-step and can be used as a tutorial. The user is encouraged to learn the FRACAS system by turning on the machine and
following along with this example.

After booting up the system (see "Getting Started", above), the user is presented with the title page, followed by the MAIN menu. The MAIN menu contains no data; it is used solely for the selection of the other screens. In this session, we will use the STORAGE page, select and use the objective screen and the four data screens, and examine the output of the model.

A.5.1 The MAIN menu.

First, examine the page. The upper left corner should identify the screen as the MAIN menu. (If not, press the ESCAPE key). The right upper corner should describe the name of the current workspace, which is the data that is now on all of the screens. At the bottom of the screen, the current major options that are available to you are displayed. In this case, the only options presented are moving the cursor and selecting an option. The cursor is moved by the arrow keys on the right side of the keyboard. Experiment with their use. Notice that the cursor can be made to move up to the workfile name. This allows the user to change the name of the file at any time.

If the diskettes have never been used, the name of this file is "empty". If not, someone has used these disks and left their data in the workspace.

Since the workspace is on disk, it is unaffected by turning off the machine. Since data in the workspace could be valuable, we should store it before beginning our own work. If your workspace file is "empty", you should follow along to see the STORAGE page. Move the cursor down to the
position marked STORAGE and press the ESCAPE key.

A.5.2 The STORAGE page.

The STORAGE page takes about thirty seconds to appear. Examine it carefully. The workspace is still defined in its usual location (although now it is called "workspace" instead of "file"), but there is an additional position under it called "filename" specifying the same name. At the bottom of the page, we see that selecting an option is done with Cntl-C, and pressing the ESCAPE key will return us to the MAIN menu.

There is a list of names down the right center of the screen. This is the catalog -- these datasets currently exist on disk, and any of them can be LOADED into the workspace, replacing the workspace's current contents. We wish to STORE the current contents of the workspace to memory. If the file already exists on disk, though, the system will not allow the STORE command to replace the file on disk. The special command REPLACE is provided for this, to prevent inadvertent loss of valuable data.

The first task is to examine the catalog and see if the current workspace name is there. If it is not select the STORE option by pressing "1". Notice that the cursor jumps to the location "filename". Since the file we wish to STORE is already there, the option is selected by pressing Cntl-C. The file is STOREd and the catalog is updated.

If the file is on the catalog, however, we must choose whether to REPLACE the file or STORE the file under a new name. If the previous user of the
system has not been identified, then it might be best to STORE the file under a new name. Press the key "I", and select a new name when the cursor jumps to the filename position. Pressing Cnt1-C will store the file and recatalog the disk. Experiment with the screen. It is frequently used and you should be familiar with it.

Before returning to the MAIN menu, LOAD the file "empty". This is the only way to clear the workspace. Press the ESCAPE key and return to the MAIN menu.

A.5.3 The Objective Screen.

Examine this screen carefully. The screen is titled in the upper left corner, as all the screens are — the workspace name is still in the upper right corner. Instructions are still on the bottom of the page. These instructions indicate that the arrow keys will move the cursor, the ESCAPE key, as usual, will return us to the MAIN menu, Cnt1-P will print the screen, and Cnt1-C will "accept" data. Accepting data means that the data on the screen is accepted by the user for storing into the workspace. Both ESCAPE and Cnt1-C will return the user to the MAIN menu, but only Cnt1-C will put the entered data into memory. Thus, ESCAPE can be used to leave a menu if the user "gets into trouble", changing data that he or she did not intend to change. The data that was shown when the screen was started will be left in memory.

The cursor started on the position asking "analyze local service?". This prompt is requesting that the user enter the periods of local service that
should be analyzed with this run. In this example, we are only interested in examining peak period local service, so the option "1" is selected for this first data entry location. The cursor automatically advances to the next position. Since no express service fits into our plans for analysis, select "4" here.

It is also necessary to indicate what services are currently being operated. Data describing these services will be entered at a later point, and used to "pivot" from in the analysis. For this reason, existing data is required in the same time periods (peak or offpeak) as the service(s) to be examined, although, for example, local data can be provided for the analysis of express services. In this example, there is both peak and offpeak local service already existing, so enter "3" for the third question and "4" for the fourth.

We next choose Objective 2 from the four possibilities. In this objective, every rider has a value to the transit operator above and beyond his or her fare. We will set a value of 50 cents at a later point to reflect our judgement that the region would be willing to support up to a 50 cent per rider deficit for new patronage. The sensitivity to this number can be tested by repeating the analysis with several different values per rider.

The other options for the objective include minimizing the deficit (Objective 1) or to maximize the ridership within the deficit limit (Objective 3). This deficit limit would be entered later in the program. By selecting Objective 4, we could specify all of the output variables and have the system report performance data on our design.
We then specify the constraints for the output variables. Since we wish the model to choose all of the variables, we enter a "1" for these data. In our case, "1" and "2" are equivalent. If we were looking at both peak and offpeak service, choosing "2" for a variable would constrain the model to pick one value for that variable that worked best for both periods. We can also prespecify a variable by entering a "3" here. In this case, we would enter the value of that variable at a later point in time.

With the screen now finished, examine it and make sure that it agrees with the data on the screen shown in figure A2. If you "accept" it, press Ctrl-C. From the MAIN menu, select the next option, the CONSTRAINTS screen. Notice that FRACAS anticipated that you would go to the next screen and moved the cursor down one step for you.

A.5.4 The Constraints Screen.

Examine this screen. The top and bottom should now be familiar. The screen itself is similar to the previous screen, but more complicated: there are 25 data items that may be specified here, and the screen accepts multi-digit real numbers, so you will have to use the <- and -> keys to move the cursor.

Not all of the data that can be entered on this screen is needed for this run, although you may put in additional information if you wish so that it will be there if needed by future analysis. To determine what is needed, you need to look at the line on the screen labelled "needed this run."
### FIGURE A2
**EXAMPLE OBJECTIVE SCREEN**

<table>
<thead>
<tr>
<th>Objectives, Constraints, Time Periods</th>
<th>File: GEORGE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time Periods &amp; Service Types:</strong></td>
<td></td>
</tr>
<tr>
<td>1 .... Analyze Local Routes?</td>
<td>1 - Peak</td>
</tr>
<tr>
<td>4 .... Analyze Express Routes?</td>
<td>2 - Offpeak</td>
</tr>
<tr>
<td>1 .... Existing Local Routes?</td>
<td>3 - Both</td>
</tr>
<tr>
<td>4 .... Existing Express Routes?</td>
<td>4 - None</td>
</tr>
<tr>
<td><strong>2 .... Objective</strong></td>
<td></td>
</tr>
<tr>
<td>1 - Minimize Deficit</td>
<td></td>
</tr>
<tr>
<td>2 - Max Weighted Riders-Deficit</td>
<td></td>
</tr>
<tr>
<td>3 - Max Riders w/Deficit Constraint</td>
<td></td>
</tr>
<tr>
<td>4 - All Predetermined</td>
<td></td>
</tr>
<tr>
<td><strong>Constraints:</strong></td>
<td></td>
</tr>
<tr>
<td>1 .... No. of Routes</td>
<td></td>
</tr>
<tr>
<td>1 .... Route Length</td>
<td></td>
</tr>
<tr>
<td>1 .... Fare</td>
<td></td>
</tr>
<tr>
<td>1 - Model Chooses, Separate</td>
<td></td>
</tr>
<tr>
<td>2 - Model Chooses, Equal (2 periods)</td>
<td></td>
</tr>
<tr>
<td>3 - Predetermined</td>
<td></td>
</tr>
</tbody>
</table>

Use (<, >) to move cursor
Cntl-C to accept data

Esc to MAIN menu
Cntl-P to print screen
Data is needed in the boxes that have a bar on this line. Data is needed for the value of rider if there is a bar above it. Notice that for this run we need only to enter data for "Fixed Costs" and for "Value of Rider". In fact, we only need to enter data for the peak/local cell for fixed costs, since this is all that we are analyzing. If we had wanted to prespecify some of the output variables, it would be done here under "Predetermined Values."

Enter "278" for fixed costs; enter "50" for the value of the rider. The screen should resemble the one in figure A3. Press Cntl-C followed by ESCAPE to move on to the next menu.

A.5.5 The Operating Screen.

To enter the next few pieces of data, it is necessary to first look at the map of the example area shown in figure A4. The corridor length is defined as the distance from the CBD via typical streets that would be used by transit routes to the edge of continuous development. Beyond this length, the street system is very incomplete, significant amounts of open land exist, and the potential for transit exists only for a few focussed clusters of development. Some judgement is needed in determining this length, since this will be the maximum route length possible within the model. Set this value to "8" miles. You will now need to press the <return> to gain control of the cursor (before the <return> is pressed, the <- is used to backspace).

Refering to the map again, notice that straight lines have been constructed
## FIGURE A3
### EXAMPLE CONSTRAINTS SCREEN

<table>
<thead>
<tr>
<th>DATA</th>
<th>Constraints, Objectives</th>
<th>Predetermined Values</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>!---------------------!</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N  L  F  t  M  Co</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>used this run</th>
<th>Number of Routes</th>
<th>Route Lngth (mi.)</th>
<th>Fare (cents)</th>
<th>No. of Trips</th>
<th>Max Deficit ($/day)</th>
<th>Fixed Costs ($/day)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; peak: offp:</td>
<td>loc exp</td>
<td>loc exp</td>
<td>loc exp</td>
<td>loc exp</td>
<td>loc exp</td>
<td>loc exp 278</td>
</tr>
</tbody>
</table>

**Value of a Rider**: 50.0 (cents)

Use (-, ->) to move cursor
Cnt1 C to accept data
Esc to MAIN menu
Cnt1-P to print screen
as the edges of the corridor. It is important that the edges are positioned so that the area between the lines is correct — routes can temporarily travel outside these boundaries. Measure the width of the corridor along typical streets between the ends of these constructed lines. This is the width of the corridor. We measure the width to be 6 miles. Enter "6", <return>.

The average walk speed is usually assumed to be 3 miles per hour. Enter 3 <return>.

The maximum distance beyond which no persons are willing to walk is based on operator experience and judgement. Typical values are around one-half mile. Some operators will assume that this value is one-quarter mile, but there are usually some transit users willing to walk beyond this distance. We use a value of .5 here in the example.

The ratio of total to in service bus hours is determined by estimating each of the parameters separately and dividing. Total bus hours is defined as the number of hours the bus is operating away from the garage. The in service bus hours are defined as the number of hours the bus is available for passenger service. We use a ratio of 1.4 in this example.

The bus operating cost varies between peak and offpeak periods. Although the model will accept a uniform cost for peak and offpeak service, it is generally better to estimate the two separately. For the peak period, we use the marginal cost of a peak-only bus, computed as the average cost per service hour over trippers, split shifts, and other driver assignments for
peak-only runs. In offpeak periods, we use the cost per service hour of a vehicle operated all day. Realize that these costs are strictly for in-service time, not including layover or deadhead times. We use a peak cost of $36/hour here. The offpeak cost is not needed.

The length of the analysis periods is the number of hours each day with peak or offpeak service. For our example, there are 4 hours of peak service each day. The length of the offpeak service per day is not needed.

The maximum load per bus reflects the equipment type and the loading standards of the property. Different equipment types can be reflected by varying the values of load per bus, cost and velocity. Since this is the maximum load is a constraint on the average load at the peak load point, it should be lower than the ultimate capacity of the vehicle. We use 40 passengers per bus in this example.

For local service, the average bus speed includes delays for boarding and alighting. The model assumes that the bus loads are near the maximum in the peak and are half of the maximum in the offpeak. The speed chosen for this example is 12 mph in the peak.

Check to see if your screen matches the one shown in figure A5. Accept the data when correct and go the Existing screen.

A.5.6 The Existing screen.

This screen allows the user to enter data for eight variables. We will
### FIGURE A5
**EXAMPLE OPERATING SCREEN**

<table>
<thead>
<tr>
<th>DATA</th>
<th>Operating</th>
<th>File: GEORGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corridor Length (mi):</td>
<td>8.0 (Y)</td>
<td>Maximum Passenger Load/Bus (s)</td>
</tr>
<tr>
<td>Corridor Width (mi):</td>
<td>6.0 (W)</td>
<td>Local</td>
</tr>
<tr>
<td>No. of Expressways:</td>
<td>1.0 (ne)</td>
<td>Express</td>
</tr>
<tr>
<td>CBD Length (mi):</td>
<td>0.5 (Lo)</td>
<td>peak: 40.00</td>
</tr>
<tr>
<td>Analysis days/year:</td>
<td>250.0 (dy)</td>
<td>offpeak:</td>
</tr>
<tr>
<td>Total/Serv. Bus-Hrs.:</td>
<td>1.40 (h)</td>
<td></td>
</tr>
</tbody>
</table>

**Bus Operating Cost**
- ($/hr) peak-only: 36.00 (6c1)
- base period: (6c3)

**Average Segment Speed (mph)**
- Local
- Express (w/stops) (w/o stops)
  - peak: 12.00
  - offpeak: |

**Length of Period (hrs)**
- for peak analysis: 4.00 (T1/60)
- for offp analysis: (T3/60)

Use <-, -> to move cursor
Cntl C to accept data
Esc to MAIN menu
Cntl-P to print screen

75
need to enter data for each variable for peak/local service only.

The number of routes is again determined from the map. Since major
branches count as routes, we count six routes.

The average route length is found by adding up the total length of all
routes counted above (including the common lengths) and dividing by the
number of routes. Short-turns are ignored in this step. If the routes are
extremely circuitous, then deviations should be omitted from the average
route length calculations. Our average route length calculation gave us 7
miles.

The average fare should be estimated for each of the service/time periods
that data is required. If the fare per service/time period is not known,
then the nominal adult fare should be used. If nominal fare is input, then
the result of the model will be nominal fare; if an average fare is input,
then the answer will be an average fare. Our fare for this example is 70
cents.

The current number of bus trips in the peak direction is calculated from the
current schedules. Short-turns here are counted as fractional values. In
this example, the number of bus trips over the six routes is 90. This is
the number of inbound trips in the morning peak plus the number of outbound
trips in the evening peak.

The total ridership over all routes is found from revenue or passenger
count data. The ridership for our example is 5,165 in both directions over
the two peaks.

The fraction of current transit riders bound to or from the CBD, including transfers in the CBD, is estimated from ridership counts or from experience. This value for the example is 0.82.

The fraction of current riders travelling in the peak direction is also determined from ridership studies or from experience. In this case, the value is 0.80, typical for peak period ridership.

The current mode share captured by transit of all CBD trips is generally obtainable by dividing transit ridership to the CBD by the total traffic to and from the CBD. The total flow of persons to and from the CBD is usually obtainable from regional transit planning agencies, state DOT’s, or downtown associations. For our purposes, we use 0.20.

Check whether your screen matched the one shown in figure A6. Then accept the data into the workspace and move to the Market screen.

The Market Screen.

This data pertains to the market characteristics of the geographic area. Along with the Existing data above, these data tend not to change much after they have been set.

The average walk speed is generally considered to be 3 mph -- this is what we use here.
### FIGURE A6
EXAMPLE EXISTING SCREEN

<table>
<thead>
<tr>
<th>DATA</th>
<th>Existing Service</th>
<th>File: GEORGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Routes (r)</td>
<td>Local peak: 6.0, offpeak:</td>
<td>Express peak: 5165, offpeak:</td>
</tr>
<tr>
<td>Route Length (mi) (u)</td>
<td>peak: 7.0, offpeak:</td>
<td>% to/from CBD (100n) peak: 82.0, offpeak:</td>
</tr>
<tr>
<td>Fare (cents) (e)</td>
<td>peak: 70.0, offpeak:</td>
<td>% in Peak Direction (100m4) peak: 80.0, offpeak:</td>
</tr>
<tr>
<td>No. of Bus Trips (t)</td>
<td>peak: 90.0, offpeak:</td>
<td>Transit CBD mkt shr % (100z) peak: 20.0, offpeak:</td>
</tr>
</tbody>
</table>

Use (-, -) to move cursor
Cntl C to accept data
Esc to MAIN menu
Cntl-P to print screen
The maximum distance beyond which no persons are willing to walk is based on operator experience and judgement. Typical values are around one-half mile. Some operators will assume that this value is one-quarter mile, but there are usually some transit users willing to walk beyond this distance. We use a value of .5 here in the example.

The average CBD parking cost is definable for both peak and offpeak users. We have used a peak CBD parking cost of $1.50, or 150 cents.

The maximum policy headway is set by the analyst based on either formal or informal service standards. These standards will not typically be binding in the peak period. We set this maximum headway to 60 minutes. Note that the specification for maximum headway must be greater than the existing headway to get a realistic validation.

The standard value for the ratio of average passenger wait time to route headway is 0.50. It can be greater than 0.50 for poorly kept, short headways and less than 0.5 for well-kept long headways. Since we expect good schedule adherence in our example, so we set this ratio to 0.40.

The sensitivities are set by examining the graphs in figure A7. Typically, the fare coefficients will differ between peak and offpeak period travel and may differ between the local and express traffic. Choose a coefficient that represents a curve in the figure that appears to represent the true fare/ridership trade-offs in the corridor being studied. You may interpolate between the curves if necessary.
SENSITIVITY CHARTS
FOR COEFFICIENTS

FIGURE A7
Alternatively, the fare coefficient can be found from the fare elasticity if it is known. The fare elasticity is the percent change in ridership at existing levels when there is a 1% change in fares from there existing levels. If $E(f)$ is the fare elasticity, then:

$$\text{fare coefficient} = \frac{E(f) \times \text{current transit mode share of CBD trips}}{\text{average fare}}$$

Using $E(f)=0.35$, we find the fare coefficient to be 0.001.

The travel time sensitivity is similarly found from either the figure or the elasticity. With $E(t)$ as the travel time elasticity, the formula for the calculation of the coefficient is:

$$\text{trav. time coef.} = \frac{E(t) \times \text{current transit mode share of CBD trips}}{\text{(average route length / 2) \times \text{average bus speed}})}$$

The value we use for the travel time coefficient is 0.003.

The walk time sensitivity can be determined by examining figure or by equation. With $E(k)$ = walk time elasticity:

$$\text{walk time coef.} = \frac{E(k) \times \text{current transit mode share of CBD trips}}{\text{min(max walk dist./walk speed; corr. width / route length) / min(max walk speed/# routes*corr. length)}}$$
The value used for the walk coefficient is 0.01.

The wait time coefficient can likewise be determined from the graphs or determined by the following equation. With a wait time elasticity of $E(w)$:

\[
\text{wait time coef.} = \frac{E(w) \times \text{current transit mode share of CBD trips}}{\langle \text{ave. wait/headway} \rangle \times \# \text{routes} \times \text{length of period/number of bus trips in peak direction}}
\]

For our purposes, we choose a wait time coefficient of 0.01.

Check to make sure your screen matches the screen shown in figure A8.
Accept the screen and run the program by selecting "Output".

The Output.

When the calculation messages have finished, the report screen will show the results. Notice that the keys that are active at this point are the arrow keys (→ & ←) which change the page that you view, the Cntl-P, which prints the report shown in figure A9 and returns the user to the MAIN menu, and ESCAPE, which returns the user to the MAIN menu without printing.

Notice the changes that were made in the system. The fare has increased from $0.70 to about $1.00, headway is reduced from 16 minutes to 13 minutes, the number of routes run comes down from 6 to 4.

These optimal values are indicative of directions which produce ridership increase, deficit decrease, or productivity increase. In this example, the
### FIGURE A8
EXAMPLE MARKET SCREEN

<table>
<thead>
<tr>
<th>DATA</th>
<th>Market</th>
<th>File: GEORGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. Walk Speed (mph)</td>
<td>3.0 (60j)</td>
<td></td>
</tr>
<tr>
<td>Max. Walk Distance (mi)</td>
<td>0.5 (H)</td>
<td></td>
</tr>
<tr>
<td>Avg. CBD Parking Cost</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(cents/trip)</td>
<td>peak: 150</td>
<td></td>
</tr>
<tr>
<td></td>
<td>offp:</td>
<td></td>
</tr>
<tr>
<td>Max. Headway Policy</td>
<td>Local Express</td>
<td></td>
</tr>
<tr>
<td>(Z) (in min)</td>
<td>peak: 60.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>offp:</td>
<td></td>
</tr>
<tr>
<td>Wait-to-Headway Ratio</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(k)</td>
<td>peak: 0.40</td>
<td></td>
</tr>
<tr>
<td></td>
<td>offp:</td>
<td></td>
</tr>
</tbody>
</table>

SENSITIVITIES
select from graphs in the manual

Local Express

Fare
pk: .001000
(a4)
offpk:

Running Time
pk: .003000
(a3)
offpk:

Walk Time
pk: .010000
(a8)
offpk:

Wait Time
pk: .010000
(a2)
offpk:

Use <, -> to move cursor

Cnt1 C to accept data

Esc to MAIN menu
Cnt1-P to print screen
<table>
<thead>
<tr>
<th></th>
<th>PEAK</th>
<th></th>
<th>OFFPEAK</th>
<th></th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Local</td>
<td>Express</td>
<td>Local</td>
<td>Express</td>
<td></td>
</tr>
<tr>
<td>Service</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&amp; Fares: No. of Routes:</td>
<td>3.75</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Route Length (mi):</td>
<td>6.76</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Avg. Headway (min):</td>
<td>12.42</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Avg. Fare (cents):</td>
<td>103.45</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Impacts: Load Per Bus:</td>
<td>40.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>40.00</td>
</tr>
<tr>
<td></td>
<td>Mode Share (CBD):</td>
<td>0.17</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Daily</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Impacts: Cost($):</td>
<td>3212.25</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>3212.25</td>
</tr>
<tr>
<td></td>
<td>Revenue($):</td>
<td>4352.77</td>
<td>0.00</td>
<td>0.00</td>
<td>4352.77</td>
</tr>
<tr>
<td></td>
<td>Deficit($):</td>
<td>-1140.52</td>
<td>0.00</td>
<td>0.00</td>
<td>-1140.52</td>
</tr>
<tr>
<td></td>
<td>User Benefit($):</td>
<td>2934.27</td>
<td>0.00</td>
<td>0.00</td>
<td>2934.27</td>
</tr>
<tr>
<td></td>
<td>Ridership:</td>
<td>4207.64</td>
<td>0.00</td>
<td>0.00</td>
<td>4207.64</td>
</tr>
<tr>
<td>Annual</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Impacts: Cost($):</td>
<td>803.06</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>803.06</td>
</tr>
<tr>
<td></td>
<td>Revenue($):</td>
<td>1088.19</td>
<td>0.00</td>
<td>0.00</td>
<td>1088.19</td>
</tr>
<tr>
<td></td>
<td>Deficit($):</td>
<td>-285.13</td>
<td>0.00</td>
<td>0.00</td>
<td>-285.13</td>
</tr>
<tr>
<td></td>
<td>User Benefit($):</td>
<td>733.57</td>
<td>0.00</td>
<td>0.00</td>
<td>733.57</td>
</tr>
<tr>
<td></td>
<td>Ridership:</td>
<td>1051.91</td>
<td>0.00</td>
<td>0.00</td>
<td>1051.91</td>
</tr>
<tr>
<td>Daily Statistics:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bus-Miles:</td>
<td>978.08</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>978.08</td>
</tr>
<tr>
<td>Bus-Hours:</td>
<td>114.11</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>114.11</td>
</tr>
<tr>
<td>No. Bus Trips:</td>
<td>72.38</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>72.38</td>
</tr>
<tr>
<td>No. Buses:</td>
<td>28.53</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>28.53</td>
</tr>
<tr>
<td>Psgr./Bus-Mile:</td>
<td>4.30</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>4.30</td>
</tr>
<tr>
<td>Psgr.-Miles:</td>
<td>14214.5</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>14214.5</td>
</tr>
<tr>
<td>Psgr.-Mile/Bus-Mile:</td>
<td>14.53</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>14.53</td>
</tr>
<tr>
<td>Avg. Travel Time (min):</td>
<td>16.89</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>Avg. Walk Time (min):</td>
<td>4.97</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>Avg. Wait Time (min):</td>
<td>4.97</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>Daily:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(in $)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cost/Passenger:</td>
<td>0.74</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.74</td>
</tr>
<tr>
<td>Revenue/Passenger:</td>
<td>1.03</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.03</td>
</tr>
<tr>
<td>Deficit/Passenger:</td>
<td>-0.27</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>-0.27</td>
</tr>
<tr>
<td>Benefit/Passenger:</td>
<td>0.70</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>-0.00</td>
</tr>
<tr>
<td>Operating Cost:</td>
<td>2934.25</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>2934.25</td>
</tr>
<tr>
<td>Fixed Cost:</td>
<td>278.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>278.00</td>
</tr>
<tr>
<td>Ratio Revenue/Cost:</td>
<td>1.36</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.36</td>
</tr>
</tbody>
</table>
value per rider is rather low (50 cents), so service is expanded only until
the deficit for the last rider reaches 50 cents. Since the deficit per
rider increases as marginal patronage is sought, most riders cost the
system less than 50 cents deficit. With a 70 cent current fare, this means
that a high revenue/cost ratio is implicitly being required. The model
suggests the best way to achieve this. Note that headways actually
improve, though routes are cut and fares increase sharply. This is a
different strategy than many systems may follow. If portions of the
solution are unacceptable, they can be constrained to acceptable values by
just returning to the Objective screen and setting the variable as
predetermined, and then moving to the Constraints page to specify a value.
The model can be rerun and the new results obtained. Using this iterative
process, the user of FRACAS should be able to achieve a better intuitive
understanding of the transit system than previously possible.
REFERENCES


