PROJECT 1

A COMPUTER METHOD FOR OBTAINING "ACTUAL" ROUTING MILEAGE IN RAILWAY NETWORKS

RUSSELL B. CROFT, CDP, BA

A PROJECT
IN
THE DEPARTMENT
OF
COMPUTER SCIENCE

Presented in Partial Fulfillment of the Requirements for
the degree of Master of Science (Applied)
at McGill University
Montreal, Canada

April, 1975

INTRODUCTION

CN has developed a highly sophisticated formula designed to indicate the cost of moving goods from one point on the rail network to another. The formula has the form $f(c_i , u_i)$ where the u's are parameters or output units which affect the cost in some way e.g. mileage, tonnage, car-days or yard-switching-minutes; and the c's are unit costs or cost per output unit , e.g. cost per mile, cost per ton, cost per car-day or cost per yard-switching-minute. Regression analysis is used to produce these unit costs. This technique tends to smooth aberrations in the actual performance statistics recorded at source, and account for them in what is considered the proper proportions for specific applications.

It is fairly easy to calculate a cost by hand for a given movement of goods shipped in a specific kind of equipment between any two pairs of points on the CN system and to do this with a fair amount of accuracy. We are able to do this even though the regression formula analyzes data summarized differently than its ultimate use. For example, in practice, it is far easier to collect and analyze the effect of all the train miles on our Southern Ontario area than it is to collect and analyze the number of train-miles from Toronto to Sarnia. Detailed data would be too costly to obtain and (for a network as large as CN's) almost prohibitive to handle in a reasonable amount of time.

Recently, a great number of requests have been made for more and more detailed costs, so that, even though the hand calculation formala is easy to use, the volume was getting out of hand. The company began to embark on wholescale profitability studies for which some sort of detailed cost was needed quickly. In addition, since the cost itself is built up by multiplying some set of unit costs by a set of performance statistics, managers submitted their requests asking that some of these performance statistics be totalled as well as the cost. At best this extra information would give only an estimate for parameters like train-miles or loaded car-miles on a particular section of track even though statistics were not readily or easily available in such detail. These requests presented a costing clerk with real problems, especially if these requests involved large geographical areas.

Consequently, a proposal was made that we should investigate some computer method to do part of the work performed by a clerk. The proposal was to produce as good a cost as possible. The costs would be compared to the revenue (information which is easy to obtain) to check the profitability. An arbitrary profitability factor was set so that any movement falling within (say) $\pm 15\%$ of break-even would be intensively recosted by hand. The purely profitable would be discarded for the time being and the purely unprofitable traffic would be forwarded to others to take some kind of remedial action.

A method to produce a satisfactory cost by computer has now been developed. The method, outlined here, has become so efficient that the natural extension was made to cost all the traffic that CN handles, and to produce various summary profitability and performance statistics for management. The process involved special techniques, not the least of which was to find some efficient way to analyze some 2,000,000 individual movements for any given year. There were several problems in this project, each one of which could be discussed in separate treatises. This paper describes only one of these problems: that of producing an acceptable mileage for use in the costing formula. The philosophy behind the method is to actually get the computer to "think like a human". As will be shown, this involves trying to simulate train movements the way our train masters actually do the same job.

SOME OF THE PROBLEMS

In the analysis of how we were going to produce a cost it was found that the cost itself could be segregated into four major components:

1) mileage related calculations--that portion of the cost directly related to the length of haul of the movement of goods (about 20 - 30% of the total)
2) tonnage related calculations--that portion of the cost directly related to the weight of goods carried (about 20 - 30% of the total)
3) engine switching related calculations--that portion of the cost directly related to the supply of empty equipment to customers and marshalling

them in yards before (and after) being hauled on trains (about 30 - 40% of the total)

4) other components including how long a railway car is in service, billing costs etc. (about 5 - 20% of the total).

It was quite evident after just a short analysis that if a good method could be found to produce an acceptable mileage, the other calculations could also be simplified. Unfortunately, there are in use three different kinds of mileage calculations, each one of which has its own purpose. These are:

1) Optimum Mileage--a theoretical mileage used to come to some "best" decision on how to handle traffic
2) Road Map Mileage--a practical mileage used to estimate the effect of a given service. It is usually the most direct mileage between two points on a well defined road.
3) Actual Mileage--an accounting mileage used by accountants to properly apportion cost data to various operations. It is the road map mileage plus any detours en route.

For costing the optimum mileage is just not applicable. Although an optimum cost based on optimum mileage may be desirable in some circumstances to indicate what we should strive for it is not really applicable when trying to find out what in fact actually did happen. On the other hand, the road map mileage was used for some time and was quite effective until it was legend

YARD	NAME
1	MONCTON
2	NEWCASTLE
3	CAMPBELLTON
4	GASPE
5	RIVIERE DU LOUP
6	FREDRICTON
7	ST. LEONARD
8	EDMUNSTON
9	JOFFRE

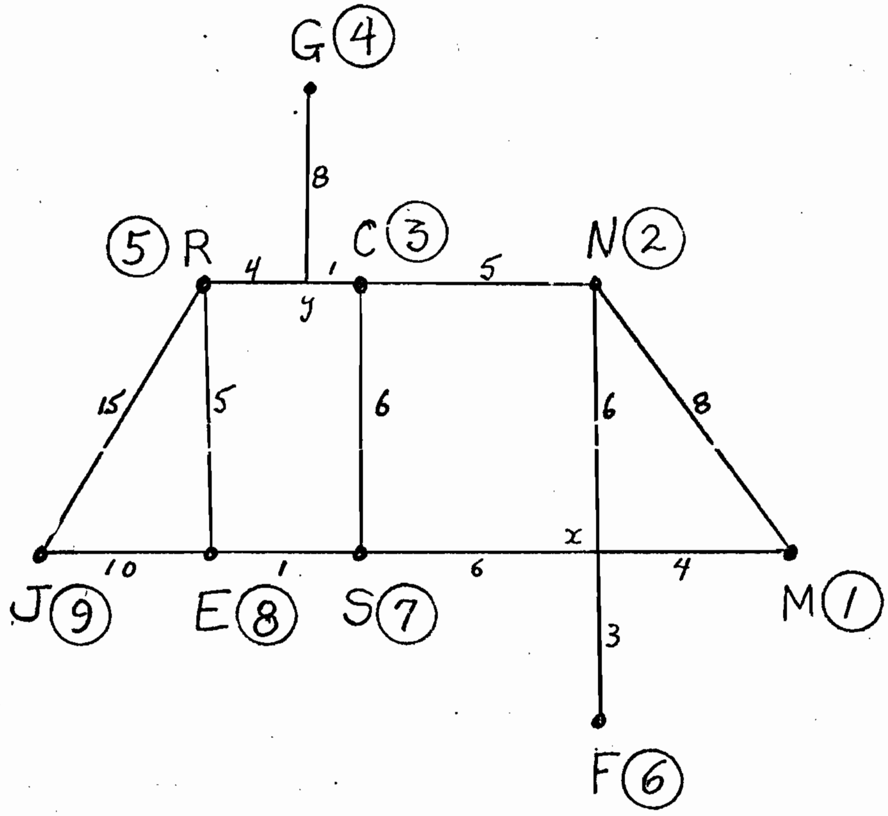

FIG. 1
Stylized Routing Map of a portion of
CN Atlantic Region

found that this kind of mileage produced too low a cost. The road map idea was originally used because that is the way a costing clerk produced his cost, and it was the original aim to come as close as possible to his calculations. In addition, road map mileage was easily inserted into a computer programme as a table and searched. As will be shown later, this method was discarded not only because it produced too low a cost, but also because it presented some special problems in computer processing. The actual mileage eventually became the concept used, but it is by no means easy to produce, computer or no computer.

THE SYSTEM - INTRODUCTORY COMMENTS

To illustrate what I mean by "actual mileage" let us consider a simplified map of a portion of CN's Atlantic Region (Fig. 1 opposite). Since this map will be used as the basis for most of this discussion it is best to understand what it represents.

The DOTS represent yards or points where traffic is delivered and marshalled.
The NUMBERS (uncircled) represent mileage between yards
The LETTERS represent the names of each yard. A legend is included.
The NUMBERS CIRCLED represent the yard code.

The orientation of the map is essentially correct: top is north and left is west. The computer is asked to follow two fundamental rules:

1) Trains must travel from yard to yard with no stopping between yards. Thus a train moving from M to S may not stop at x. If it did, this would imply that x itself should be a yard.

2) Trains must travel "westward". We must assume that any route from M to R is exactly the same route from R to M but in the reverse direction.

This last rule can be followed easily by having the yard numbers increase from east to west. We will ask the computer to strive to find a route from a low numbered yard to a higher numbered yard. If this is not the case, we can reverse the yard numbers to make it the case. For example a request for a route from G to M (i.e. from 4 to 1) will be calculated from M to G (1 to 4) because the two routes are the same.

To develop a mileage we ask the computer to trace a route from yard to yard, tallying the mileage at the same time. Thus if we ask the computer to give a mileage from M to C the computer answers "M to N (8 miles), N to C (5 miles), for a total of 13 miles". But what happens if the computer is asked to give a mileage from G to R? Is the route G to R for a tally of 8 + 4 = 12 miles or is it G to C then to R for a tally of 8 + 1 + 1 + 4 = 14 miles? The first option we might call the "road miles" defined earlier, while the second (if it occurs) will be the actual miles described earlier. In practice, the route from G to R is actually G to C to R. This kind of routing happens often on the CN system, so it must be tallied exactly if and when it happens because it does affect the cost. One can easily see that the difference in the mileage-related cost using the two different mileages (assuming the cost is directly proportional to mileage) will be about 14%.

THE SYSTEM - A SIMPLE EXAMPLE

Now that we have our ideas fixed, let us try to find a method to define routes for this kind of network. In practice, when a railway car is at M, the trainmaster asks himself the following questions:

1) Where is this railway car going?
2) On which train must I place this railway car to get it from "here" to "there"?

Let us suppose that the railway car in question is to move from M to R. The train master answers his questions this way:

1) The railway car is going to R
2) From M there are three possible train services (arcs on the network):
   a) M to N
   b) M to S
   c) M to F
3) Of these M to F is no good because F is a "dead end". Thus my choice is one of the other two, but which one?
4) What do my train orders say? (this is a set of operating rules defined from past experience)
5) The rules state:
   a) if a railway car is at M and is going to any yard whose code is less than or equal to 5, then send it on a train going from 1 (i.e. M) to 2 (i.e. to N).
   b) if a railway car is going to the yard coded 6, then send it on a train going directly to 6.

c) all other railway cars are to be placed on trains going from 1 (i.e. M) to 7 (i.e. to S).

Thus the train is sent on to N from where similar train orders tell the local yard master to route our train hauling the railway car to C and then to R.

"Aha!", cries a sharp observer. "Surely this is not the right route. Suppose, rather than taking the route M-N-C-R (total 18 miles) that the train master sends the train from M to S then to E to R for a total of 16 miles?" As stated earlier, herein lies the whole tale. It turns out that the service on the R-E line is slow and predominantly serves E from R. Thus to save delays, and to satisfy the customer's desire to get traffic delivered quickly, the railway car is sent by a different route. The computer system that we are asked to design must take into account all these kinds of problems, so that the mileage the computer is asked to recreate will be the mileage that the train masters actually design. This route will be the "actual route" which results in tallying the "actual mileage" we defined earlier.

There is another example of this kind of actual service which must be given. Observe the routes F-N-C and F-S-C. The train service states:

Monday and Friday	take F-N-C and return the next day to F via the same route
Wednesday	take F-S-C and return the next day to F via the same route.

Now if a railway car is at S available for routing on Thursday going to C, rather than wait until next Tuesday for the train to go in that direction (F-S-C) the train orders can state that the railway car be picked up on Thursday, delivered to F (Thursday) then take Friday's train to C. Thus the route would be S-F-N-C for a total of 6 - 3 - 3 - 6 = 23 miles rather than the 6 miles from S to C. This situation does not occur often but when it occurs we would like to be able to handle it easily.

It turns out that the train orders that the train master uses are quite explicit and can be translated easily into a computer programme. The whole idea will be to obtain a mileage which represents as closely as possible the way railway cars actually move most of the time. Let us assume that the complication of alternate service due to time of week does not occur for our sample map. We are now in a position to examine how such a set of operating rules can be translated into computer jargon.

I have devised a scheme for numbering the yards, which, briefly, follows these rules:

1) Identify the main line. It is numbered last.
2) Yards for a continuous set of lines (arcs) must be numbered in sequence.
3) Only yards are numbered. Intersections like X and Y are called dummy yards.
4) The eastern-most yard of a set of lines (arcs) is numbered first.
5) Numbering continues westerly, stopping at a dead-end or at another east-west line, until numbering can proceed no further.
6) When there is more than one un-numbered line leading from a yard, or dummy yard, number them in order by length (number of lines or arcs) with the dead-ends being numbered first.

To number our sample Map 1 procede as follows:

A) M-S-E-J is the main line

B) M gets 1 (it is eastern-most)

C) I continue numbering at N (it gets 2) because the other line is the main line.

D) X does not get numbered. I must stop numbering this line here and continue to C; it gets 3.

E) I do not number S (it is on another east-west line)

F) I continue at Y; Y is not numbered; G gets 4 (a dead end)

G) I continue to R; it gets 5; E and J are not numbered because they are on another east-west line.

H) I return to M; then to X (not numbered); F gets 6; then S gets 7; E gets 8; and J gets 9.

We shall see that the routing programme depends on this numbering scheme.

THE ROUTING ALGORITHM*

To generalize, we must change our terminology only slightly. Because "yard" connotes special conditions in the ultimate computer programme, we actually call the dots in the sample map JUNCTIONS. For our sample map, then, let us make a table of all the possible train runs (network arcs) using the following rules:

1) the table has four columns:

   in column 1 place the "from" or ORIGIN JUNCTION for each arc. Call it "OJ".

   in column 2 place the "to" or DESTINATION JUNCTION for each arc. Call it "DJ".

   in column 3 place the highest junction number that can be accessed by going from the Origin Junction to the Destination Junction. Call it "LJ" for limit junction. For example, in the previous discussion we found, according to the train orders, that the highest junction we could reach by going from 1 to 2 was 5; and from 1 to 6 was 6. (We will see later this number can be manipulated in any way that we please to make the computer programme do what we want it to do!!)

   in column 4 place the miles between OJ and DJ. Call it "MI"

2) Arrange the table so that column 1 is in order. This is necessary because when there is more than one arc leaving a junction, this creates extra entries in the table to represent these arcs. Later, this rule, too, must be altered.

* Strictly speaking this is an heuristic. At this stage of its development, we cannot prove that the procedure is exhaustive. Thus "algorithm" will be used in a very general way.

OJ	DJ	LJ	MI
1	6	6	7
1	7	9	10
1	2	5	8
2	6	6	9
2	3	9	5
3	4	4	9
3	7	7	6
3	5	9	5
4	5	9	14
5	8	8	5
5	9	9	15
6	7	9	9
7	8	9	1
8	9	9	10

FIG. 2
Routing Table for Fig. 1

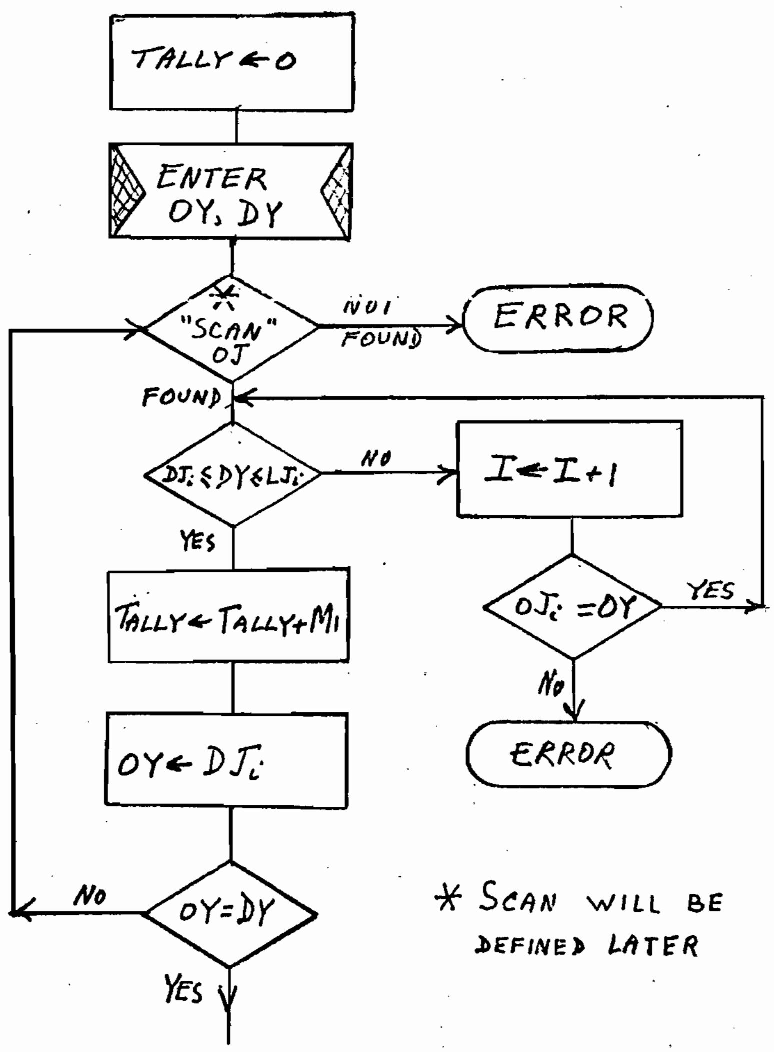

FIG. 3
Routing Algorithm Flow Chart

The result appears in fig 2 opposite. Figure 3 shows a sample algorithm flow chart that will allow us to use this table to find a mileage. In words this algorithm states:

1) start the mileage at 0.
2) enter a pair of numbers representing the "from" or Origin Yard (OY) and the "to" or Destination Yard (DY).
3) Scan the first column until we find the first OJ number equal to the Origin Yard. Call this number OJ_i. (Scan will be defined later)
4) If <u>not</u> $DJ_i \leqslant DY \leqslant LJ_i$ then repeat this step with i = i + 1 as long as OJ_{i+i} still equals OY, otherwise stop and process an error routine. (We should have listed all possible arcs, if we leave one out, this error routine will tell us so.)
5) If $DJ_i \leqslant DY \leqslant LJ_i$ then a) tally the miles MI_i and
   b) change OY to read DJ_i
6) If the new OY equals DY then stop. The mileage has been tallied; the algorithm terminates. Otherwise repeat steps 3 through 6.

Using the algorithm, let us try to find the mileage from M to G (i.e. from yard 1 to yard 4). I will form a summary table and leave the reader to find his way through it.

STEP	OY	DY	TALLY	i	TESTS	REMARKS
1	--	--	0			Set tally to zero
2	1	4	0			OY = 1; DY = 4 Routing begins
3	1	4	0	1		The first link in the table is found
4	1	4	0	1	no good	DY not in DJ - LJ range for line 1; increase i by 1; OY = new OJ
4	1	4	0	2	ok	DY now in DJ - LJ range for line 2
5	2	4	8			Change OY to new DJ; tally miles
6	2	4	8		continue	New OY ≠ DY, so continue
3	2	4	8	4		4th line of table is found
4	2	4	8	4	no good	DY not in DJ - LJ range for line 4; increase i by 1; OY = new OJ
4	2	4	8	5	ok	DY now in Dj - LJ range for line 5
5	3	4	13			Change OY to new DJ; tally miles
6	3	4	13		continue	New OY ≠ DY, so continue
3	3	4	13	7		7th line in table is found
4	3	4	13	7	ok	DY is in DJ - LJ range for line 7
5	4	4	22			Change OY to new DJ; tally miles
6	4	4	22		quit	New OY = DY so quit; the mileage is 22

OJ	DJ	LJ	MI
1	6	6	7
1	7	9	10
1	2	5	8
2	6	6	9
2	3	9	5
3	4	4	9
*3	5	5	5*
3	7	7	6
*3	2	6	5*
3	5	9	5
*4	3	9	9*
5	8	8	5
*5	3	7	5*
5	9	9	15
6	7	9	9
7	8	9	1
8	9	9	10

FIG. 4
Routing Table for FIG. 1
(Revised)

Our sharp-eyed observer finds another fault: try to find a mileage from R to S. The junction numbers are from 5 to 7 (i.e. from East to West) - a perfectly reasonable request. But in this case there is no entry in the table that will allow us to get junction 7 between a DJ and an LJ. Our observer is quite correct: I left it out to illustrate another point. It will be remembered that we formulated two basic rules for the computer to strive to use when finding a route. One of these (rule 2) stated that the computer should route from a low numbered junction to a higher if possible. Unfortunately this may not be possible all the time. Even in practice, a train must take a backward step just to get further ahead. In the same way we can get the computer to do the same thing. Notice that there is no logic in the algorithm which says that the low-high order must be followed. Thus we are able to insert into the table a line which reads "5 3 7 5" which would be analogous to a train order saying: "if a railway car is at 5 destined for yards 6 or 7 (yards greater than 5), then send it on a train going to 3". From yard 3 (i.e. C), the normal low-high (East - West) routing will be resumed.

It is not obvious with this small table, but the order in which arcs from a specific node are listed is of great importance. To maintain an East - West (low yard number to high yard number) order the rule (2) for ordering the table must be altered. We will want the shorter East - West runs to occur first (these are the "simple movements"). Then the West - East segments, followed by the "long haul" segments. When this guide-line is followed, the ammended routing table takes the form shown in Fig 4. The amended lines are marked for convenience. Note that the old segment from 4 to 5 has been changed to reflect more closely the route actually taken.

As an exercise, I ask our astute observer to verify that the route from R to F (i.e. from 5 to 6) using the new table, is 19 miles; and, the route from G to R (i.e. from 4 to 5) is 14 miles. Note that although it appears that the route from R to F is from West to East geographically, by the definition of the yard numbering scheme we can consider this an east-west movement, ie., a movement from a low numbered function to a high numbered function.

PROGRAMMING CONSIDERATIONS

It is time to summarize the above scenario. We have produced a mileage which represents the way a train master would route trains. The algorithm to do this is simple and easy to programme. But what we have not been able to establish is whether the algorithm is efficient. What can we say about the algorithm if the table has 500 or 5000 lines in it? How can such a table be searched efficiently? If standard search techniques are used, would not the search time be prohibitive? Let us analyze these problems before coming to any conclusions.

Throughout computer science literature it has been shown that one of the best methods to search an ordered in-core table is to use a BINARY SEARCH technique in some form. If a BINARY SEARCH method is used to search this table of N lines there will be $K \cong \text{Log}_2 N-1$ comparisons to find the first occurence of an origin yard, plus some "L" sequential tests after that to decide which one of several equal destination junctions to use. This process will be repeated for each of "I" iterations for each route desired. If there are "R" routes for which we need mileages then the total timing in comparisons would be in the order of:

$$T \text{ (comparison)} \simeq R\ I\ (\log_2 N-1 + L)$$

where N = Number of table lines (elements)

L = Number of sequential searches

I = Number of iterations for 1 route

R = Number of routes

At CN, our table of arcs has (typically) 1000 arcs, i.e. N $\simeq$ 1000; there are on the average 2 sequential searches i.e. L = 2

it takes on the average 20 iterations to find a route: I = 20

Using these figures and inserting some sample number of routes gives

for R =	100	1000	10,000	100,000	routes
T =	24K	240K	2.4M	24M	comparisons

(K = 1,000 M = 1,000,000)

These figures indicate that if we are not careful, this method may take some time. However, in practice, we at CN capitalized on the fact that there are less than 1000 junctions for the entire rail network map, enabling us to use 3-digit codes for the junction numbers. The table itself has

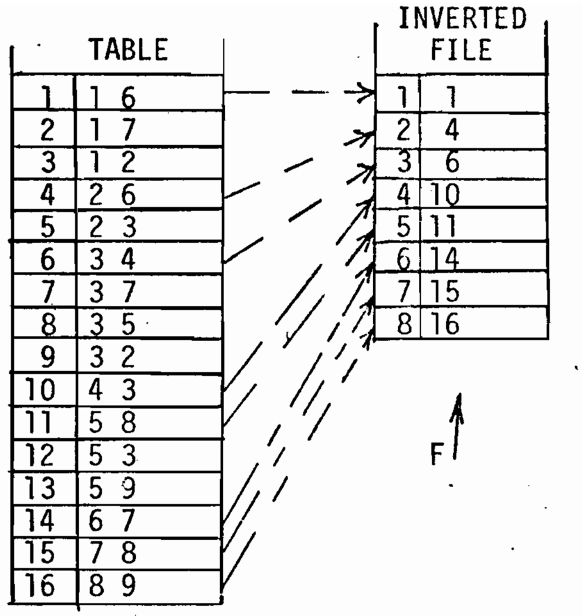

FIG. 5

Organization for the Routing Table

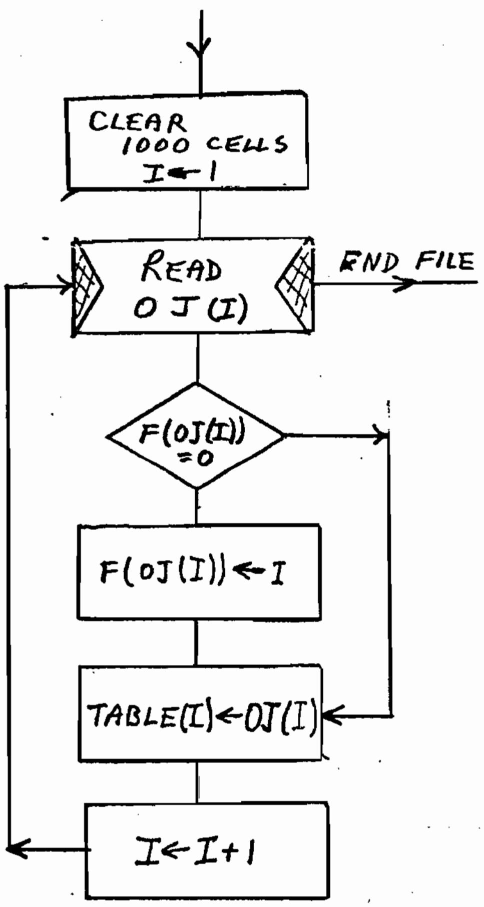

FIG. 6

Flow Chart to Build Routing Table

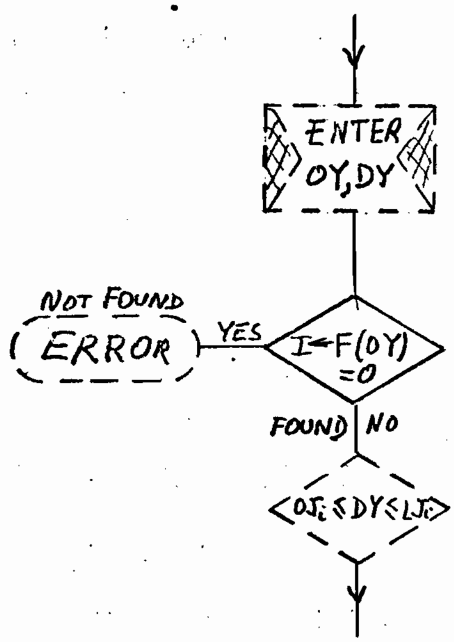

FIG. 7

Routing Table Search Algorithm

close to 1000 lines. All we had to do was to keep track of the first occurance of each junction in the table, and record these positions in another table. In this manner we produced a "mini-inverted file" in core to help reduce search time, i.e., we listed the positions of the first occurance of each junction separate from the table itself. This allowed us to use a "double subscripting" technique (a very fast index register operation) to find where to start searching in the table during routing rather than using the Binary Search. (Fig. 6 shows the algorithm to build both these tables; the results of the algorithm is shown in Fig. 5; and Fig 7 shows the actual table-search flow chart stated only as " SCAN OJ " in the flow chart of Fig 3. These charts have been inserted for interest's sake.) Thus the $\log_2 N-1$ factor can be eliminated from the formula for T and replaced by just 1 comparison resulting in a 75% reduction in the timing:

for	R =	100	1000	10,000	100,000	records
	T =	6K	60K	600K	6M	comparisons

(K = 1,000 M = 1,000,000)

For a typical batch run of about 75,000 routings, using a moderately fast computer, these timings translate to about 1 to 1.5 hours of processing. This may sound like a lot of time, but considering the accuracy of the cost that this mileage produces, and considering that comparable costs cannot be obtained for this much traffic in a reasonable amount of time, the investment in computer time is will spent.

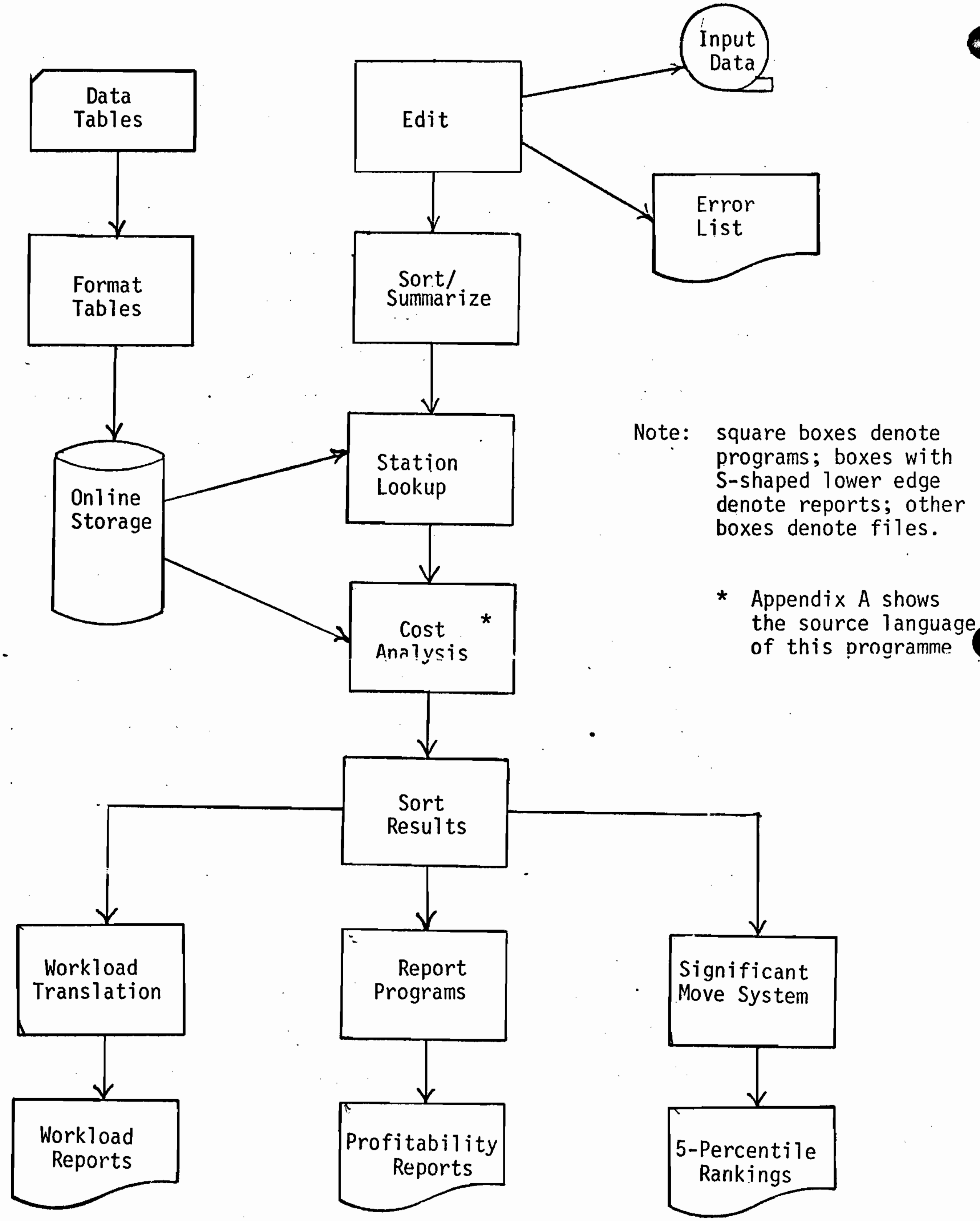

Fig. 8

The Complete Cost Analysis System

IMPLEMENTATION AT CNR

Actually the mileage obtained from this method does not tell the whole story. There are about 5500 stations on the CN rail network of which less than 10% are classified as yards or junctions. The routing and mileaging algorithm described here is only a small section, but the most important section, of a series programmes which has been designed to examine and highlight different aspects of the costs and profitability of all CN traffic. The system is a powerful planning tool that allows us to analyse large volumes of revenue traffic information, and then to produce concise and meaningful summary reports of this information upon which action may be taken. Some of this action might be:

A) Forming equipment lease and purchase strategies based on the profitability of certain railway car types;

B) Examining cerrtain marketing policies based on commodity cost and/or profitability characteristics;

C) Determining train service feasibility; or

D) Determing rail line abandonment feasibility.

The cost analysis system can be broken into four logical sections:

1) Editing, modifying, and grouping of input data.

2) The detailed analysis of the cost, of which the routing algorithm is a part.

3) The organization, ranking and grouping of the output data.

4) The production of reports.

Fig. 8 (opposite) shows complete programme flow of the cost analysis system.

Input to the system is from the commodity detail summary file, a file containing revenue information for all the traffic CN handles in any given month. Of course any source of data can be used as long as the data contains certain information necessary to produce a cost. The specific information needed to calculate a cost of a movement is:

1) The originating station number (eg. 14522 represents M (Moncton) in figure 1).

2) The destination station number (these two parameters define the route that a railway car takes-the route calculated by the algorithm described here).

3) A traffic code - a code that describes how much of the movement occurred on the CN territory. These codes are:

   LF - local forwarded (100% CN)

   IF - interline forwarded (terminated outside CN)

   IR - interline received (originated outside CN)

   BR - bridge received (originated and terminated outside CN)

4) A railway car type code-used to calculate equipment costs (eg. 250 is a standard box car)

5) A commodity code-used to calculate-commodity related cost (eg. 733 is the code for cloth and fabrics)

6) The number of cars to which the above information applies

7) The total tonnage for the above information

8) The total revenue, included to calcuate the profitability.

The input file containing the above information represents some 2.5 million carloadings each year summarized to about 600,000 records by the time it reaches the edit phase of the cost analysis system. The editor first checks for compatability of codes then formats each record ready for sorting. The types of code checking performed include:

A) Compatability between commodity codes and equipment types, eg., we want to remove records showing livestock travelling in tank cars, or bulk liquid petroleum travelling in box cars, etc.

B) Equipment carrying excessive weight (most of these are decimal point misplacements)

C) Impossible station codes

D) Records showing impossible revenues

After these checks and reformating the new file is ordered by ascending sequence by origin and destination station codes, the result of which is passed to the cost analysis programmes proper.

The cost analysis section is divided into two segments. Although the two segments perform one logical step, and could be run together, hardware and procedure constraints at CN have necessitated the division into two parts. As pointed out earlier, there are about 5500 stations on the CN rail network of which less than 10% are yards or junctions. The first of these segments calculates a mileage for moving this traffic on local or 'way-freight' trains to the yards. This is a fairly simple but not trivial procedure. This has

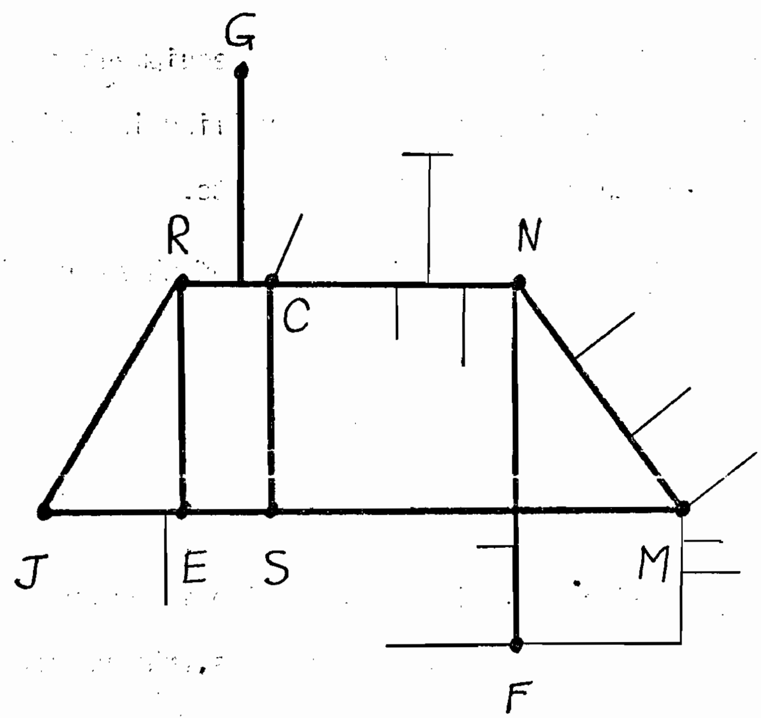

FIG. 9

Stylized Detail Map of a portion of
CN Atlantic Region

the effect that the extra trackage, representing simple one-arc routes, can be eliminated from the total network leaving a skeleton network of about 500 junctions and their associated arcs or routes. As an example, fig 9 opposite shows the detailed map of the portion of the Atlantic Region used as an example for the discussion of the routing algorithm. The second of these parts is the costing programme of which the routing algorithm described here is part.

The computations in the cost analysis programmes depend on data which is stored in different tables in the system. These data tables contain the following information:

1) Unit costs, i.e. cost/mile or cost/train-mile, etc.

2) Freight car descriptions i.e., tare weights, or cost/day

3) Commodity costs, i.e. cost/freight claims

4) Empty movement of freight car probabilities

5) A station number table

6) Train performance data; and

7) The junction table described earlier.

The first of the cost anlaysis programmes has a very simple function--to search the 5500 element station table, assign the junction code for the routing algorithm, and to calculate the mileage from the stations at each end of the movement to the junctions. The second part is divided into three sections. The first section calculates the cost of the local or 'way-freight' train service from the stations at each end of the movement to its associated junction. These costs are computed from information in the data tables, and information contained in the input record: The second section is the routing algorithm. As each arc of the route is found a cost is calculated for that route using much the same calculations as in the first part and using much the same information. The third section is the totalling section where all the costs are added, including fixed costs, such as billing and cleaning costs. Accumulated at the same time are some performance statistics such as total train-miles and total car-days. Each input record is costed and mileaged in this way, producing an output file ready for profitability analysis.

Upon completion of the cost analysis run, the output file is passed to a number of utility programmes, from which various summaries and reports are produced. Most of these reports are produced only on request, eliminating unnecessary computer processing and storage of volumes of little-used paper. Service on these requests is fast, and depending on the complexity of the request, can be completed within 24 hours. The costed file is normally kept as a historical record and stored on magnetic tapes indefinitely.

As a final presentation I have included a sample of the various kinds of reports that are produced from the Cost Analysis System.

Fig. 10 - Some costed movements of commodity 733 (cloth, fabrics) between various stations on the rail network. For example, the last line shows a movement from St. Henri (Montreal) to Winnipeg

of one car weighing 22 tons (commodity weight) costing $xxx# based on 1345 miles. The mileage was calculated by the routing algorithm. The other columns represent figures which are needed for planning purposes.

Fig. 11 - A summary of all the movements of commodity 733 for a given time period. The last line of figure 1 is included in the top line of this example. Note that the traffic has been separated between profitable, unprofitable and suspect (marginal) traffic. The mileage in this case is the weighted average mileage for all 88 cars.

Fig. 12 - A summary of the regenerated workload statistics (costing parameters) by segment of track. As mentioned earlier, these kinds of statistics are not generally maintained by the accounting system because they would be too costly to obtain or record. For example, the 3rd group of lines from the top shows the section of track from Coteau, Quebec to Glen Robertson, Ontario. This section of track was used for the movement described in figures 10 and 11. These are estimates, and, as it turns out, fairly good ones. When all these statistics are added to give a grand total for the entire CN system, the deviation from the data that can be collected is not more than 5-10%, depending on the statistic.

Figures 10, 11, 12, 13 which follow are internal confidential reports, consequently all cost and revenue related figures have been deleted.

Fig. 13 - A sample of the significant moves system. These reports are ordered by total parameter (in this case by tons and by route). This page of the report shows the third 5-percentile group. The total line shows that only 42 records (point to point movements) have accounted for 15% of total system tonnage, of which the 23 listed records are part. Incidentally, the report shows that the 42 records represent about 7% of the total car miles, a figure generated by the routing algorithm.

The above reports admittedly place much emphasis on the costs that can be derived from the system. Mileage has a direct effect on about 20-30% of the cost and an indirect effect on about another 30%. The mileage itself is the actual mileage or mileage that reflects every time that a wheel on a railway car turns. Thus the real point is that all these costs would never have been possible if we could not have come up with some simple and acceptable method to produce an accurate mileage between pairs of points on the CN rail network. Many times, the simplest solution seems to be the best solution, and in this case I have been able to demonstrate that there is a very simple method to produce the mileage and thus to produce a cost.

**TESTING PROGRAM 073311 R E S E A R C H & D E V E L O P M E N T C O S T I N G S E R V I C E S ** AUG. 12/74

GROUP = 3.50

COMMODITY = 733 CLOTH , FABRICS N.O.S.

T C	ORIGN STATN	DESTN STATN	OA	CARS	TONS	REVENUE SURS IN	COST	NET REVENUE	TON PER CAR	REV PER CAR	CST PER CAR	NREV PER CAR	MILES	REVENUE PER TON	NREV PER TON	REV PER NTM	REV/CST RATIO	CAR TYPE
LF	12605	33273	62	1	22				22				798					220
	HALIMPEX	MONYARDPQ																BOX STL 40F8FDR60T
LF	12605	33273	62	2	66				33				798					225
	HALIMPEX	MONYARDPQ																BOX STL 40F9F DR 60T
LF	12605	33360	62	1	25				25				798					225
	HALIMPEX	COTSTPAUI																BOX STL 40F9F DR 60T
LF	12605	33376	62	1	20				20				798					220
	HALIMPEX	PTSTCHARR																BOX STL 40F8FDR60T
LF	12605	33376	62	3	53				17				800					510
	HALIMPEX	PTSTCHARR																INSUL BOX 40F STD
LF	27724	43340	65	1	23				23				370					100
	STHYACINT	NEWTORONT																BOX FOREIGN
LF	27724	43340	65	1	23				23				370					110
	STHYACINT	NEWTORONT																BOX STL 40F8FDR CUF60T
LF	27724	43340	65	2	47				23				370					115
	STHYACINT	NEWTORONT																BOX STL 40F9FDR CUF60T
LF	27724	43340	65	24	563				23				370					220
	STHYACINT	NEWTORONT																BOX STL 40F8FDR60T
LF	27724	43340	65	33	768				23				370					225
	STHYACINT	NEWTORONT																BOX STL 40F9F DR 60T
LF	27724	43630	65	1	22				22				441					225
	STHYACINT	CLARKSON																BOX STL 40F9F DR 60T
LF	33128	44510	66	1	20				20				387					210
	MONEAST	HAMILTON																BOX STL 50F DDR 80T
LF	33128	46210	66	1	34				34				389					210
	MONEAST	WATERLOON																BOX STL 50F DDR 80T
LF	33128	46210	66	1	31				31				389					240
	MONEAST	WATERLOON																BOX STL 50F DDR 70T
LF	33170	42230	66	1	12				12				331					225
	MONMORSTR	WTORONTO																BOX STL 40F9F DR 60T
LF	33170	42230	66	1	12				12				331					530
	MONMORSTR	WTORONTO																INSUL BOX 50F STANDARD
LF	33170	42510	66	1	12				12				331					530
	MONMORSTR	TORCHEST																INSUL BOX 50F STANDARD
LF	33170	42572	66	1	12				12				331					540
	MONMORSTR	ORIOLE																INSUL BOX 50FCMP 12FDR
LF	33270	94110	66	1	12				12				2890					900
	STLAUREPQ	VICTORIBC																MISCELLANEOUS CARS FGN
LF	33324	64210	66	1	22				22				1345					225
	STHENRI	WINNIPEG																BOX STL 40F9F DR 60T

NOTE: ALL CONFIDENTIAL INFORMATION HAS BEEN DELETED

FIG. 10

Sample Commodity (detail) Cost Report

R E S E A R C H & D E V E L O P M E N T C O S T I N G S E R V I C E S

**TESTING PROGRAM 073311 ** AUG. 12/74

GROUP = 3.50 * SINGLE COM. SUB-GROUP *

COMMODITY = 733 CLOTH , FABRICS N.O.S.

*** CCM-271 TOTAL ***

T C	RCDS	CARS	TONS	REVENUE SUBS IN	COST	NET REVENUE	TON *---PER	REV CST CAR---*	NREV	MILES *-----PER	REVENUE TON-----*	NREV	REV PER NTM	REV/CST RATIO
PROFITABLE TRAFFIC														
LF	25	88	2080				24			610				
IR	16	20	301				15			264				
BF	8	29	545				19			405				
TO	49	137	2926				21			536				
UNPROFITABLE TRAFFIC														
LF	3	3	36				12			331				
IF	1	1	10				10			0				
IR	10	20	214				11			473				
BF	1	1	7				7			547				
TO	15	25	267				11			438				
SUSPECT TRAFFIC														
TOTAL TRAFFIC														
LF	28	91	2116				23			606				
IF	1	1	10				10			0				
IR	26	40	515				13			351				
BF	9	30	552				18			407				
TO	64	162	3193				20			528				

FIG. 11

Sample Commodity (Summary) Profitability Report

NOTE: ALL CONFIDENTIAL INFORMATION HAS BEEN DELETED

SUMMARY BY LINK SEP. 15/73 YEAR 1972 FSF- CT235

OPERATIONS AND MAINTENANCE WORKLOADS

FORECAST YEAR 1973
COMMODITIES-ALL
CAR TYPES -ALL

FROM	TO	MILES		LOADED CARS	EMPTY CARS	NET TONS	LOADED CAR MILES	EMPTY CAR MILES	NTM (000)	GTM (000)	CHANGE 1971 TO 1973 TOTAL CARS AMOUNT	PCT	TOTAL TONS AMOUNT	PCT
HEPSTJC	NAKINA	143	W	190	37	5960	27170	5291	852	1445	10104	0.0	0	0.0
			E	37	190	956	5291	27170	137	252	1762	0.0	0	0.0
			T	227	227	6916	32461	32461	989	1697	11867	0.0	0	0.0
COTEAU	CORNWAL	30	W	14442	10508	541836	435152	315240	16362	25870	862333	0.0	0	0.0
			E	11639	0	269034	350380	0	8124	15785	526166	0.0	0	0.0
			T	26031	10508	810870	785532	315240	24486	41655	1388500	0.0	0	0.0
COTEAU	GLENROB	15	W	2642	0	67506	40519	0	1046	1938	129200	0.0	0	0.0
			E	5869	2701	270234	90347	40519	4188	6171	411400	0.0	0	0.0
			T	8511	2701	337740	130866	40519	5234	8109	540600	0.0	0	0.0
CORNWAL	BRCKVIL	58	W	14113	10251	521383	820389	594600	30343	48249	831879	0.0	0	0.0
			E	11384	0	258902	661470	0	15067	29509	508775	0.0	0	0.0
			T	25497	10251	780285	1481859	594600	45410	77758	1340655	0.0	0	0.0
CORNWAL	COTEAU	30	W	276	125	15913	8317	3759	481	662	22066	0.0	0	0.0
			E	139	0	3150	4178	0	95	187	6233	0.0	0	0.0
			T	415	125	19063	12495	3759	576	849	28300	0.0	0	0.0
BRCKVIL	KINGSTN	40	W	14117	10373	519990	573799	414929	21214	33770	844250	0.0	0	0.0
			E	11367	0	257877	461069	0	10520	20630	515750	0.0	0	0.0
			T	25484	10373	777867	1034868	414929	31734	54400	1360000	0.0	0	0.0
BRCKVIL	CORNWAL	58	W	250	102	14252	14534	5949	829	1147	19775	0.0	0	0.0
			E	114	0	1818	6618	0	106	250	4310	0.0	0	0.0
			T	364	102	16070	21152	5949	935	1397	24086	0.0	0	0.0
BRCKVIL	KINGSTN	40	W	52	2	687	2100	109	28	74	1850	0.0	0	0.0
			E	3	0	168	121	0	7	10	250	0.0	0	0.0
			T	55	2	855	2221	109	35	84	2100	0.0	0	0.0
KINGSTN	BELVILL	47	W	14133	10286	522082	672217	483465	24902	39598	842510	0.0	0	0.0
			E	11340	0	256644	538565	0	12241	24032	511319	0.0	0	0.0
			T	25473	10286	778726	1210782	483465	37143	63630	1353829	0.0	0	0.0
KINGSTN	BRCKVIL	46	W	250	104	14252	11667	4790	667	922	20043	0.0	0	0.0
			E	114	0	1818	5295	0	85	201	4369	0.0	0	0.0
			T	364	104	16070	16962	4790	752	1123	24413	0.0	0	0.0
KINGSTN	BELVILL	47	W	53	40	707	2511	1906	34	89	1893	0.0	0	0.0
			E	45	0	2084	2127	0	99	146	3106	0.0	0	0.0
			T	98	40	2791	4638	1906	133	235	5000	0.0	0	0.0

FIG. 12

Sample Maintenance Workloads Report by Track Segment

NOTE: ALL CONFIDENTIAL INFORMATION HAS BEEN DELETED

***
***
***
*****
***
*

	RECORDS		PCT	REVENUE	PCT R	CAR LDS	PCT C	TONS	PCT T	NET REV	PCT N	CARMILES	PCT M	CAR DAYS	PCT D	
LF	64118	33338	103			255		18393				684420		3870		15
LF	37154	42435	755			354		17454				169876		19		15
LF	77420	60512	23			371		17242				678188		4135		15
IF	35140	35160	773			494		16537				39520		3453		15
IR	55954	55838	759			177		16490				60534		969		15
IF	33376	35106	23			393		16148				30654		2391		15
IR	55903	55838	23			182		16081				29120		961		15
LF	48460	45300	23			556		16038				29468		4707		15
IR	55978	55838	11			173		15836				58824		947		15
IR	55954	55600	103			241		15623				147974		1609		15
IR	47960	42310	43			226		15341				21271		1314		15
IF	55838	45110	21			743		14938				392824		5734		15
IR	55978	55838	21			547		14692				132525		2931		15
IR	55954	43325	89			872		14487				898160		6552		15
BF	35130	35160	773			402		14460				45852		1144		15
LF	51480	14789	15			370		14260				665298		4192		15
IR	55978	55600	763			325		13963				192962		2081		15
IR	55951	55600	23			152		13318				92162		1075		15
IR	55690	55600	103			187		13302				71808		952		15
IF	55838	55517	773			444		13175				102128		2801		15
IR	55954	55838	21			385		12986				107046		2076		15
IF	35106	35160	773			340		12886				21760		2092		15
IF	77420	93330	15			304		12802				477750		2546		15
LF		16	0.07		3.14	9202	4.48	471788	6.58		1.86	9689418	3.74	87744	4.40	
IF		11	0.05		3.06	15933	7.76	361190	5.04		1.89	6554609	2.53	108267	5.42	
IR		14	0.06		1.29	4960	2.42	234389	3.27		0.37	2960038	1.14	31544	1.58	
BF		1	0.00		0.04	402	0.20	14460	0.20		0.04	45852	0.02	1144	0.06	
TOT		42	0.19		7.54	30497	14.86	1081827	15.09		4.15	19249917	7.43	228699	11.46	

FIG. 13

Sample 5-Percentile Ranking Report

NOTE: ALL CONFIDENTIAL INFORMATION HAS BEEN DELETED

APPENDIX A

SOURCE LISTING OF

COST ANLAYSIS PROGRAMME

```
IDENTIFICATION DIVISION.
PROGRAM-ID.        '07340000'
AUTHOR. D          RB CROFT, CDP
INSTALLATION.      CN RESEARCH AND DEVELOPMENT.
DATE-WRITTEN.      SEPTEMBER 1974.
DATE-COMPILED.     APR  3, 1975.
REMARKS.           THIS PROGRAM IS THE COST ANALYSIS PROGRAM
                   OF THE CN COST RESEARCH SECTION COSTING SYSTEM.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT INPUTFILE     ASSIGN UT-S-IN01.
    SELECT TABLES-FILE   ASSIGN UT-S-IN02.
    SELECT OUTPUTFILE    ASSIGN UT-S-OUT01.

DATA DIVISION.
FILE SECTION.
FD  INPUTFILE
    RECORDING MODE IS F
    RECORD CONTAINS 80 CHARACTERS
    BLOCK CONTAINS 0 RECORDS
    LABEL RECORDS ARE STANDARD
    DATA RECORD IS COSTINPUT.
01  COSTINPUT.
    02  PART-03.
    05  ORG-DEST-IN            PIC X(10).
    02  PART1.
    05  STATION-O              PIC 9(5).
    05  LIM-STN-1-O            PIC 9(5).
    05  LIM-STN-2-O            PIC 9(5).
    05  JCT-1-O                PIC 9(4).
    05  JCT-2-O                PIC 9(4).
    05  MILES-1-O              PIC 999V9.
    05  MILES-2-O              PIC 999V9.
    05  TRAIN-SUB-O            PIC 9(4).
    02  PART2.
    05  STATION-D              PIC 9(5).
    05  LIM-STN-1-D            PIC 9(5).
    05  LIM-STN-2-D            PIC 9(5).
    05  JCT-1-D                PIC 9(4).
    05  JCT-2-D                PIC 9(4).
    05  MILES-1-D              PIC 999V9.
    05  MILES-2-D              PIC 999V9.
    05  TRAIN-SUB-D            PIC 9(4).
FD  TABLES-FILE
    RECORDING MODE IS F
    RECORD CONTAINS 35 CHARACTERS
    BLOCK CONTAINS 0 RECORDS
    LABEL RECORDS ARE STANDARD
    DATA RECORDS ARE SEPARATOR.
01  SEPARATOR.
    05  TABLE-SEPARATOR        PIC X(4).
    05  FILLER                 PIC X(31).
01  TABLEB.
    05  ENTRY-BB               PIC X(20).
    05  FILLER                 PIC X(15).
01  TABLEBB.
    05  OJBB         COMP      PIC S9(4).
    05  FILLER                 PIC X(33).
FD  OUTPUTFILE
    RECORDING MODE IS F
    RECORD CONTAINS 14 CHARACTERS
    BLOCK CONTAINS 0 RECORDS
    LABEL RECORDS ARE STANDARD
    DATA RECORD IS COSTED-OUTPUT.
01  COSTED-OUTPUT.
    05  ORIG-DEST-OUT          PIC X(10).
    05  MILES-OUT              PIC 9999.
```

```
WORKING-STORAGE SECTION.
01  GENERAL-GARBAGE.
    05  FILLER                  PIC X(8) VALUE 'WORKSTOR'.
    05  SW-1  COMP-3            PIC S9.
    05  DIR-SW  COMP-3          PIC S9 VALUE ZERO.
    05  DIRECTION-CD COMP-3     PIC 9.
    05  SAME-LINK-SW COMP-3     PIC S9 VALUE +0.
    05  TOTAL-MILES-OUT         PIC S9(5)V9.
    05  NOT-USED                PIC S99.
01  DIRECTION-DETERMINANT.
    05  DIRECTION               PIC 99V9.
    05  DIR-CD REDEFINES        DIRECTION.
        10  W-PRT               PIC 99.
        10  D-PRT               PIC 9.
01  SNAP-CON DISPLAY            PIC XXXX VALUE 'SNAP'.
01  SNAP.
   02  SNAP-AREA                USAGE IS COMPUTATIONAL-3.
    05  OJ                      PIC S9(3).
    05  OJ1 COMP-3              PIC 9(3).
    05  OJ2 COMP-3              PIC 9(3).
    05  DJ1 COMP-3              PIC 9(3).
    05  DJ2 COMP-3              PIC 9(3).
01  SUBSCRIPTS                  COMPUTATIONAL.
    05  M                       PIC S999.
    05  A                       PIC S999.
    05  SUB                     PIC S9(4).
    05  PASS-NO                 PIC S9(4).
    05  ITT                     PIC S999.
    05  N                       PIC S9999.
    05  COST-AS-THRU-SW         PIC S9.
    05  COST-AS-WAY-SW          PIC S9.
    05  WAY-LIMIT               PIC S9(4).
01  COUNTERS                    COMPUTATIONAL.
    05  TOTAL-RECS              PIC S9(10) VALUE ZEROS.
01  ITERATION-CTR  COMP-3       PIC S9(10) VALUE ZERO.
01  FILLER                      PIC X(8) VALUE 'GEN WORK'.
01  GENERAL-WORK-AREAS          COMPUTATIONAL-3.
    05  O-D-CONV.
        10  C-STATION-O         PIC 99999.
        10  C-STATION-D         PIC 99999.
    05  ASSIGN-SW               PIC 9.
01  FILLER                      PIC X(8) VALUE 'TRIPWORK'.
01  TRIP-WORK-AREAS             COMPUTATIONAL-3.
    05  TOTAL-MILES             PIC S9(5)V9.
01  ND-C-SAVE.
    05  ND-CTRS-SAVE COMP-3 OCCURS 91 TIMES PIC S999.
01  DIFFERENCES.
    05  DIFF1       COMP        PIC S9(9).
    05  FILLER      REDEFINES   DIFF1.
        10  DIFF1-SIGN          PIC X.
        10  FILLER              PIC XXX.
    05  DIFF2       COMP        PIC S9(9).
    05  FILLER  REDEFINES       DIFF2.
        10  DIFF2-SIGN          PIC X.
        10  FILLER              PIC XXX.
01  TABLE-B-THROUGH-INFO.
    05  JUNCTIONB       COMP    PIC S9(4).
    05  TRIP-DIRECTION          PIC S9(4) COMPUTATIONAL.
    05  COST-CODE               PIC X.
    05  DISPLAY-ERROR.
        10  ERROR-CODE          PIC 9.
```

```
          10  DISPLAY-ORG       PIC 999.
          10  FILLER            PIC 99 VALUE ZERO.
          10  JUNCTIONA         PIC S9(4) COMPUTATIONAL.
          10  REGIONA           PIC S9(4) COMPUTATIONAL.
          10  DIRECTIONA        PIC S9(4) COMPUTATIONAL.
          10  SWITCH-CLNA       PIC 999V99 COMP-3.
 01   TABLES-AREA.
      05  TBL-A-ENTRY.
          10  STATS-FROM-STATION-TBLA.
              15  WAY-MILES     PIC 9(4)V9 COMP-3.
      05  TBL-B-ENTRY REDEFINES TBL-A-ENTRY.
          10  STATS-FROM-ROUTING-TBLB.
              15  MILESB        PIC 9(4)V9  COMP-3.
 01   ROUTING-TABLES.
    02  TBLB-CON   DISPLAY      PIC XXXX VALUE 'TB20'.
    02  TABLE-B.
      05  ENTRY-B          OCCURS       625 TIMES.
    02  ROUTING-TABLEB     REDEFINES    TABLE-B.
      05  ENT-B            OCCURS       625  TIMES.
          10  OJB          COMP         PIC  S9(4).
          10  DJB          COMP         PIC  S9(4).
          10  LJB          COMP         PIC  S9(4).
          10  TRNB         COMP         PIC  S9(4).
          10  RATIOCDBB    COMP         PIC  S9(4).
          10  REGION-B     COMP         PIC  S9(4).
          10  MIB          COMP-3       PIC  9(4)V9.
          10  THSWB        COMP-3       PIC  99V999.
          10  HOURS-B      COMP-3       PIC  99V9.
 01   POINTERS COMPUTATIONAL.
      05  TBX OCCURS 397 TIMES          PIC S999.
      05  O-T-TBL-POINTER OCCURS        362 TIMES PIC S999.
```

```
      ****************************************************************
      *              SECTION 1 - TABLE BUILDER                       *
      ****************************************************************
           EJECT
       PROCEDURE DIVISION.
           MOVE LOW-VALUE TO POINTERS.
           MOVE 0 TO A.
           OPEN INPUT TABLES-FILE.
           MOVE ZERO TO N.
       BEGIN-TABLES.
           MOVE 1 TO SUB.
       READ-IT.
           READ TABLES-FILE AT END GO TO TABLES-END.
       DUMMY-STATE.
           MOVE ZEROS TO NOT-USED.
       READ-IT-EXIT.
           IF TABLE-SEPARATOR EQUAL TO 'TBLB'
           PERFORM READ-IT
           GO TO READ-ROUTING-TABLE
           ELSE GO TO READ-IT.
       READ-TABLES.
           READ TABLES-FILE         AT END GO TO TABLES-END.
       BRANCH-PARAG.
       READ-ROUTING-TABLE.
           IF    TABLE-SEPARATOR EQUAL TO 'TBLA',
           THEN GO TO TABLES-END.
           MOVE ENTRY-BB TO ENTRY-B (SUB).
           IF    TBX (OJBB) = ZERO,
           THEN MOVE SUB TO TBX (OJBB),
                ADD 1 TO A,
                MOVE A TO O-T-TBL-POINTER (OJBB).
           ADD 1 TO SUB.
           GO TO READ-TABLES.

       TABLES-END.
           CLOSE TABLES-FILE.
           OPEN INPUT INPUTFILE, OUTPUT OUTPUTFILE.
           MOVE ZEROS TO ERROR-CODE.
      ****************************************************************
      *              SECTION 2 - READ AND INITIALIZE                 *
      ****************************************************************
       READ-RECORD.
           READ INPUTFILE       AT END GO TO END-OF-JOB.
           ADD 1 TO TOTAL-RECS.
       PRE-PROCESS.
           PERFORM INITIALIZE-RECORD THRU INITIALIZE-EXIT.
           PERFORM BEGIN-PROCESSING THRU EXIT-ROUTING.
           ADD ITT TO ITERATION-CTR.
       END-COSTING.
           PERFORM FINAL-FORMULAE THRU FINAL-FORMULAE-EXIT.
           GO TO READ-RECORD.
       NODE-ZERO.
       INITIALIZE-RECORD.
           MOVE STATION-O TO C-STATION-O.
           MOVE STATION-D TO C-STATION-D.
           PERFORM NODE-ZERO VARYING M FROM 1 BY 1 UNTIL M IS
               GREATER THAN 91
           MOVE ZEROS TO TOTAL-MILES-OUT
                         SAME-LINK-SW, DIRECTION-CD,
                         M,
                         ASSIGN-SW,
                         ERROR-CODE,
                         COST-AS-THRU-SW,
                         COST-AS-WAY-SW,
                         TRIP-DIRECTION,
           SW-1.
       INITIALIZE-EXIT.
           EXIT.
```

```
*****************************************************************
*              SECTION 3 - ALGORITHM                            *
*****************************************************************
 BEGIN-PROCESSING.
         IF ZERO EQUAL TRAIN-SUB-O OR TRAIN-SUB-D
             GO TO TEST-LIMIT-STATIONS.
         IF   (TRAIN-SUB-O EQUAL TRAIN-SUB-D
              AND JCT-1-O EQUAL JCT-1-D
              AND JCT-2-O EQUAL JCT-2-D)
         OR   (JCT-1-O EQUAL JCT-1-D
              AND JCT-1-O EQUAL JCT-2-D
              AND JCT-2-O NOT EQUAL JCT-2-D)
                 NEXT SENTENCE
         ELSE GO TO TEST-LIMIT-STATIONS.
                 NOTE  THAT THE ABOVE TEST CHECKS FOR 2 STNS
                     ON THE SAME LINK.
         COMPUTE WAY-MILES = MILES-1-D - MILES-1-O.
         MOVE JCT-1-O TO OJ1, DJ1, JUNCTIONA.
         MOVE 2 TO PASS-NO.
         MOVE 1 TO SAME-LINK-SW.
         MOVE TRAIN-SUB-D TO SUB.
         PERFORM SWITCH-WAY-COST-RTN THRU CONVERT-EXIT.
         MOVE ZERO TO ITT.
         GO TO ARE-WE-FINISHED-ROUTING.
 TEST-LIMIT-STATIONS.
         IF   STATION-D IS GREATER THAN LIM-STN-2-O,
         OR   STATION-D IS LESS THAN LIM-STN-1-O,
         THEN MOVE MILES-2-O TO WAY-MILES,
              MOVE JCT-2-O TO OJ1, JUNCTIONA,
         MOVE TRAIN-SUB-O TO SUB,
              MOVE 1 TO PASS-NO,
              PERFORM SWITCH-WAY-COST-RTN THRU CONVERT-EXIT,
         ELSE MOVE MILES-1-O TO WAY-MILES,
              MOVE JCT-1-O TO OJ1, JUNCTIONA,
              MOVE 1 TO PASS-NO,
              ADD 1 TO TRIP-DIRECTION,
         MOVE TRAIN-SUB-O TO SUB,
              PERFORM SWITCH-WAY-COST-RTN THRU CONVERT-EXIT,
              ADD 1 TO TRIP-DIRECTION.
         MOVE 999 TO OJ2.
         MOVE JCT-1-D TO DJ1.
         MOVE JCT-2-D TO DJ2.
         MOVE  ZERO TO ITT.
         IF   OJ1 IS GREATER THAN DJ1,
         AND OJ1 IS GREATER THAN DJ2,
         THEN GO TO MOVE-AND-REVERSE.
 ARE-WE-FINISHED-ROUTING.
         ADD 1 TO M.
         ADD 1 TO ITT.
         IF   ITT IS GREATER THAN 80,
         THEN GO TO ERROR-3.
         IF   OJ1 IS EQUAL TO DJ1,
         THEN PERFORM FINISHED-ROUTING-1,
              GO TO EXIT-ROUTING.
         IF   OJ1 IS EQUAL TO DJ2,
         THEN PERFORM FINISHED-ROUTING-2,
              GO TO EXIT-ROUTING.
 SHOULD-WE-REVERSE.
         MOVE TBX (OJ1) TO N.
         IF   N = 0,
         THEN GO TO ERROR-9.
 CHECK-ELEMENT.
         IF   OJ1 IS GREATER THAN DJ2,
         THEN GO TO TEST-DJ1.
```

```
          IF    OJ1 IS GREATER THAN DJ1,
          THEN GO TO MOVE-AND-REVERSE.
      INTERVAL-TEST.
          COMPUTE DIFF1 = DJ1 - DJB (N).
          COMPUTE DIFF2 = DJ1 - LJB (N).
          IF    DIFF1-SIGN IS NOT EQUAL TO DIFF2-SIGN,
          OR DJ1 = DJB (N)
          OR DJ1 = LJB (N)
          GO TO PRE-COST-RTN.
      *   THE ABOVE CHECKS IF DJ1 IN TABLE RANGE
          COMPUTE DIFF1 = DJ2 - DJB (N).
          COMPUTE DIFF2 = DJ2 - LJB (N).
          IF DIFF1-SIGN IS NOT EQUAL TO DIFF2-SIGN
          OR DJ2 = DJB (N)
          OR DJ2 = LJB (N)
          PERFORM FINISHED-ROUTING-2;
          MOVE DJ2 TO DJ1
          MOVE 999 TO DJ2
          ELSE GO TO ADJUST-RTN.
      * THE ABOVE CHECKS IF DJ2 IS IN TBL RANGE
      PRE-COST-RTN.
          MOVE DJB (N) TO OJ1, JUNCTIONB.
          MOVE MIB (N) TO MILESB.
          MOVE TRNB (N) TO SUB.
          IF    OJB (N) IS GREATER THAN DJB (N),
          THEN ADD 1 TO TRIP-DIRECTION,
          MOVE 1 TO DIR-SW.
          NOTE  THIS IS THE ENTRY TO THE COST ROUTINE.

          PERFORM THROUGH-COST-MODULE THRU THROUGH-COST-EXIT.
          IF    DIR-SW IS EQUAL TO 1,
          THEN ADD 1 TO TRIP-DIRECTION,
               MOVE ZERO TO DIR-SW.
          GO TO ARE-WE-FINISHED-ROUTING.
      FINISHED-ROUTING-1.
          IF    SAME-LINK-SW = 1,
          THEN GO TO EXIT-ROUTING.
          IF    DJ2 IS EQUAL TO 999,
          THEN GO TO EXIT-ROUTING.
          MOVE MILES-1-D TO WAY-MILES.
          MOVE JCT-1-D TO JUNCTIONA.
          MOVE TRAIN-SUB-D TO SUB
          MOVE 2 TO PASS-NO.
          PERFORM SWITCH-WAY-COST-RTN THRU CONVERT-EXIT.
      FINISHED-ROUTING-2.
          MOVE MILES-2-D TO WAY-MILES.
          MOVE TRAIN-SUB-D TO SUB
          MOVE 2 TO PASS-NO.
          PERFORM SWITCH-WAY-COST-RTN THRU CONVERT-EXIT.
      TEST-DJ1.
          IF    OJ1 IS GREATER THAN DJ1,
          THEN PERFORM FINISHED-ROUTING-2,
               MOVE DJ2 TO DJ1,
          PERFORM REVERSE
          ELSE PERFORM FINISHED-ROUTING-1.
          MOVE 999 TO DJ2.
          GO TO SHOULD-WE-REVERSE.
      ADJUST-RTN.
          ADD 1 TO N.
          IF    OJ1 IS EQUAL TO OJB (N),
          THEN GO TO CHECK-ELEMENT.
          MOVE ZERO TO OJ.
      ERROR-8.
          GO TO READ-RECORD.
```

```
ERROR-7.
             GO TO READ-RECORD.
ERROR-2.
      GO TO READ-RECORD.
ERROR-3.
      GO TO READ-RECORD.
ERROR-9.
      GO TO READ-RECORD.
MOVE-AND-REVERSE.
      IF    DJ2 IS EQUAL TO 999,
      THEN PERFORM REVERSE,
             GO TO SHOULD-WE-REVERSE,
      ELSE PERFORM FINISHED-ROUTING-1,
             MOVE 999 TO DJ2,
             PERFORM REVERSE,
             GO TO SHOULD-WE-REVERSE.
REVERSE.
      ADD 1 TO M.
      MOVE OJ1 TO OJ.
      MOVE DJ1 TO OJ1.
      MOVE OJ TO DJ1.
      MOVE OJ2 TO OJ.
      MOVE DJ2 TO OJ2.
      MOVE OJ TO DJ2.
      ADD 1 TO TRIP-DIRECTION.
      ADD 1 TO M.
EXIT-ROUTING.
      EXIT.
```

```
*****************************************************************
*                SECTION 4 - WAY-FREIGHT COSTING                *
*****************************************************************
 SWITCH-WAY-COST-RTN.
     IF    SUB IS NOT EQUAL TO ZERO,
     THEN NEXT SENTENCE,
     ELSE GO TO YARD-RTN.
     COMPUTE DIRECTION = TRIP-DIRECTION / 2.
     IF    D-PRT IS GREATER THAN ZERO,
     THEN MOVE 2 TO DIRECTION-CD,
     ELSE MOVE 1 TO DIRECTION-CD.
     GO TO BRANCH-LINE-RTN.
 SUBSIDY-CHECK.
 YARD-RTN.
     GO TO SWITCH-WAY-EXIT.
 BRANCH-LINE-RTN.
     COMPUTE TOTAL-MILES-OUT = TOTAL-MILES-OUT + WAY-MILES.


     IF    SUB IS GREATER THAN WAY-LIMIT,
           MOVE JUNCTIONA TO JUNCTIONB,
           MOVE 1 TO COST-AS-THRU-SW,
           PERFORM THRU-LOOP THRU THROUGH-COST-EXIT,
           GO TO SWITCH-WAY-EXIT.


 WAY-FRT-RTN.
 WAY-RATIO-LOOP.
 SWITCH-WAY-EXIT.
 CONVERT-EXIT.
     MOVE ZERO TO COST-AS-THRU-SW.
*****************************************************************
*                SECTION 5 - THRU-FREIGHT COSTING               *
*****************************************************************
 THROUGH-COST-MODULE.
     COMPUTE DIRECTION = TRIP-DIRECTION / 2.
     IF    D-PRT IS GREATER THAN ZERO,
     THEN MOVE 2 TO DIRECTION-CD,
     ELSE MOVE 1 TO DIRECTION-CD.
     ADD MILESB TO TOTAL-MILES-OUT.
 THRU-LOOP.
 THROUGH-COST-EXIT.
     EXIT.
 FINAL-FORMULAE.
     ADD .5 TO TOTAL-MILES-OUT.
     MOVE ORG-DEST-IN TO ORIG-DEST-OUT.
     MOVE TOTAL-MILES-OUT TO MILES-OUT.
     WRITE COSTED-OUTPUT.
 FINAL-FORMULAE-EXIT.
     EXIT.
*****************************************************************
*                SECTION 8 - CLOSE FILES                        *
*****************************************************************
 END-OF-JOB.
     EXHIBIT NAMED TOTAL-RECS.
     CLOSE INPUTFILE, OUTPUTFILE.
     GOBACK.
```