ED 042 367 24v EM 008 423

AUTHOR	Newton, Rita M.; Thomas, Warren H.
TITLE	Developing a Computer Program for Bus Routing. Final Report.
INSTITUTION	State Univ. of New York Research Foundation, Albany.
SPONS AGENCY	Office of Education (DHEW), Washington, D.C. Bureau of Research.
BUREAU NO	BR-9-B-124
PUB DATE	Jul 70
GRANT	OEG-2-9-420128-1062(010)
NOTE	139p.
EDRS PRICE	EDRS Price MF-$0.75 HC-$7.05
DESCRIPTORS	*Bus Transportation, *Computer Programs, *Scheduling, School Buses, Student Transportation

ABSTRACT

A computer-based method was developed that can translate available data about schools, students, ~~and bus~~ facilities into a set of bus routes and schedules prior to the start of the school year. Each route can be so designed via the computer model that student riding time and bus capacity constraints are satisfied at the same time that total bus travel (including running empty) and number of routes required to service all the stops are minimized. The mathematical models developed were programed in FORTRAN IV for use on a CDC 6400 computer and were applied to four schools. An efficient routing system involving six possible bus route origins and 96 stops was developed for one of these schools in 61 seconds using a CDC 6400 computer. A bibliography and program listing are appended. (Author/MF)

ED042367

FINAL REPORT

Project No. 9-B-124

Grant No. OEG-2-9-420128-1062(010)

DEVELOPING A COMPUTER PROGRAM FOR BUS ROUTING

Rita M. Newton
Warren H. Thomas
State University of New York at Buffalo
Buffalo, New York 14214

The Research Foundation of the
State University of New York
P.O. Box 7126
Albany, New York 12224

July 1970

U.S. DEPARTMENT OF
HEALTH, EDUCATION AND WELFARE

Office of Education
Bureau of Research

EA 008 423

FINAL REPORT

Project No. 9-B-124

Grant No. OEG-2-9-420128-1062(010)

DEVELOPING A COMPUTER PROGRAM FOR BUS ROUTING

Rita M. Newton
Warren H. Thomas
State University of New York at Buffalo
Buffalo, New York 14214

July 1970

The research reported herein was performed pursuant to a grant with the Office of Education, U.S. Department of Health, Education, and Welfare. Contractors undertaking such projects under Government sponsorship are encouraged to express freely their professional judgment in the conduct of the project. Points of view or opinions stated do not, therefore, necessarily represent official Office of Education position or policy.

U.S. DEPARTMENT OF
HEALTH, EDUCATION AND WELFARE

Office of Education
Bureau of Research

ABSTRACT

This report describes and evaluates a practical computer based method for translating data concerning:

1. the location of each school to be serviced by a bus fleet,
2. the locations and numbers of students to be transported to each school,
3. the time interval during which the students are to be transported, and
4. the available bus facilities

into a set of bus routes which specify school-to-school sequencing of each bus and the stop-to-stop route to be followed in traveling to every school. Each route is designed in such a way that the bus capacity and student riding time constraints are satisfied while attempting not only to minimize the total bus travel time (including running empty) for a school but also to minimize the number of routes required to service all the stops associated with the school. The mathematical models developed were programmed in FORTRAN IV for use on a CDC 6400 computer and were applied to four schools in the Williamsville New York Central School District. An efficient routing system involving 6 possible bus route origins and 96 stops was developed for one of these schools in 61 seconds on a CDC 6400 computer.

ACKNOWLEDGEMENTS

This report is based on the dissertation of Mrs. Rita M. Newton entitled "Bus Routing in a Multi-School System" completed as part of the requirements of the Ph.D. program in Operations Research at the State University of New York at Buffalo. This program is administered by the Department of Industrial Engineering.

The effectiveness of this project was greatly enhanced by the cooperation and enthusiastic support of John Allan, assistant to the superintendent of schools, and Charles Bastian, director of transportation, of the Williamsville, New York Central School District. Appreciation is also due to the Board of Cooperative Educational Services, First Supervisory District of Erie County which provided all the key-punching services for this project.

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

I. INTRODUCTION

School districts are aware of the power of the digital computer as a tool for both reducing cost and improving service in the management of their educational programs. The computer has already proven its worth in financial accounting, in personnel administration, in class scheduling, and in planning school construction.

One area of school administration which has not yet been adequately served by the capability of the computer is the management of the transportation system. Since the cost of procurement, maintenance, and operation of a bus fleet requires a large portion of a school district's budget, the director of transportation is expected to minimize these costs while simultaneously providing an acceptable level of service.

At present, most schools prepare bus routes and schedules manually by using a large map of the district and a listing of the school census. Since a single school may have as many as two hundred bus stops and a district as many as twenty-five schools, this procedure is not only time consuming but requires an excessive amount of administrative talent which could be better utilized in other endeavors. Moreover, the quality of the bus routes prepared by hand is a function of the scheduler's experience, i.e. the best routes are usually prepared by the most experienced schedulers.

In addition to the bus scheduling problem, school administrators are also interested in being able to evaluate the sensitivity of current fleet operations to various changes. For example, it would be useful to be able to easily examine how population fluctuations, a change in bus fleet size, a change in school boundaries, or a change in policy concerning the maximum allowable walking distance would affect bus operations.

Therefore, a great need exists for some means for generating school bus routes and schedules efficiently.

This dissertation describes and evaluates a computer based methodology for translating data concerning:

1. the identification and location of each school which is to be serviced by the bus fleet,
2. the location and number of students to be transported to each school,
3. the time interval during which students are to be transported, and
4. the available bus facilities

into a set of bus routes which specify school-to-school sequencing of each bus and the stop-to-stop route to be followed in traveling to the school. Each route is designed in such a manner that the bus capacity and maximum allowable student riding time constraints are satisfied while attempting to not only minimize the total bus travel time (including running empty) for a school but also to minimize the number of routes required to service all the stops associated with the school.

The output of the computer program based upon this methodology is the detailed information needed to prepare specific bus schedules and bus passes for each student. In addition it provides a means for determining overall measures of schedule performance as total travel time, average riding time, average bus load, etc. These performance measures coupled with the inexpensive and rapid computer development of a complete set of bus routes for a school district make feasible a quantitative evaluation of the effect of any administrative policy changes on the transportation system.

The computer model makes operationally feasible the generation of bus routes and schedules immediately prior to the start of the school year when knowledge of transportation demands is most accurate. Moreover, it provides a tool for evaluating alternative new school sites in a district experiencing significant population growth.

II. LITERATURE REVIEW

A. School Bus Routing Problem

A review of the literature concerned with school transportation reveals that most of the studies conducted in this area deal with standards for school buses, methods of allocating funds to school districts for transportation purposes, surveys of the status and philosophy of school transportation in various geographic regions, and examination of legislation affecting school transportation. The absence of extensive literature dealing with the school bus routing problem indicates that either interest in this problem area is relatively recent or that a large portion of the attempts at developing a general method for school bus routing have proved to be fruitless and have therefore not been reported.

Dantzig, Fulkerson, and Johnson[12] gave the allusion that the problem of using a digital computer to design school bus routes was simple and straightforward. Intrigued by this, Boyer[5] attempted to solve a five stop school bus problem by applying the simplex algorithm. Because the application of a linear programming technique to the problem seemed to be highly impractical, he developed the Sequential Steps Method[5]. This procedure is based upon the premise that all students should be transported the shortest possible distance to school. Each route starts at the bus stop which is the farthest distance from the school and proceeds along the shortest path to the school in such a manner that no isolated bus stops are created and that a bus stop is assigned to only one route.

Thompson[34] tested the effectiveness of the Sequential Steps Method in determining school bus routes for a hypothetical school. He learned that although student riding time was lowered, this method tended to generate more routes for a school than other manual methods favored by the transportation personnel participating in the study. Although this method was sequential, it was not amenable to computer programming. Moreover, it made no provision for imposing bus capacity and passenger riding time constraints.

Boyer[6,7] described a procedure for designing school bus routes in which a set of bus routes or a route system is developed manually. A computer is then used to list all possible ways in which the stops of a bus route can be visited. By inspection, the best permutation for each individual route of the route system is determined. After several arbitrary route systems have been analyzed in this manner, the best route system is selected.

Since this method requires a great deal of manual work and personal judgement, it offers little improvement over the current manual methods of designing bus routes and schedules. Moreover, this method makes no provision for imposing a passenger riding time constraint.

Tillman[35] applied the technique of dynamic programming to the school bus routing problem in which all routes start and end at the school. In this formulation stage j was the number of the bus being loaded or the number of the route being designed, the decision variable was the number of stops made by bus j or the number of stops assigned to route j, the stage input was the number of stops not yet serviced at stage j, and the return obtained at each stage was the minimum number of miles traveled by bus j. The objective was to minimize the sum of the returns subject to a bus capacity constraint. Using this method, an optimum solution was obtained for a problem involving five bus stops, three buses, and forty students.

Although dynamic programming guarantees an optimal solution and allows the interstop travel time matrix to be asymmetric, it is an impractical method because of the extremely large number of calculations which have to be performed for even relatively small school bus routing problems.

Newton and Thomas[29,30] described a practical method for generating school bus routes and schedules by computer. Given the matrix of interstop travel times, which may be asymmetric, bus routing is accomplished by a two step procedure. First, a single near-optimal route which starts at the school, visits every stop once, and terminates at the school is determined. This route, the solution of the traveling-salesman problem associated with the given set of bus stops, is then partitioned into individual bus routes which satisfy bus capacity, bus loading policy, and passenger riding time constraints. The order of the route determined in step one is preserved during the partitioning process and all routes originate and terminate at the school. This heuristic procedure has been used to solve an eighty stop problem in approximately six minutes on a 7090 computer.

Davis[15] described a branch and bound algorithm for solving the school bus routing problem in which the interstop travel time matrix is symmetric, bus capacity and passenger riding time constraints are imposed, and all routes start and end at the school. This procedure partitioned the set of all possible routes into mutually exclusive subsets of routes by solving a series of transportation problems. The cost of each subset was the total traveling time required by the optimum solution to the transportation problem associated with the subset.

Although this method is amenable to computer programming and guarantees an optimal solution, it proved to be an impractical one. For example, an attempt to solve a thirteen stop bus problem, whose interstop travel time matrix was symmetric, subject only to a bus capacity constraint from[14] was abandoned when no feasible solution was obtained after approximately eighteen minutes CDC 6500 time had been spent on the problem.

B. Delivery Problem

Considerably more information is available on the closely related delivery problem[1,8,9,10,14,16,18,21,36]. The delivery problem is concerned with the determination of routes for a vehicle, initially located at a depot, which visits a number of delivery or pickup points and returns to the depot. Since the capacity of the vehicle is less than the total quantity of goods which must be transported, several trips must be made. The delivery problem usually is not subject to a traveling time or distance constraint whereas the school bus routing problem is nearly always constrained by both bus capacity and maximum allowable student riding time.

Balinski and Quandt[1] formulated the delivery problem as an integer programming problem. Although this method guarantees an optimal solution, it is undesirable because of the large number of variables and constraints required to express a delivery problem involving relatively few stops. Moreover, the available integer programming algorithms are often unable to achieve solutions to even moderately large problems even though theoretically they should always determine the optimum solution.

Hayes[21] applied the branch and bound method of Little, Murty, Sweeney, and Karel[27] to the delivery problem. This procedure partitioned the set of all possible routes into mutually exclusive subsets by either assigning or not assigning a particular link to a route. The cost of each subset was the minimum possible total length of any set of routes containing the link assignment associated with the subset.

Although this method guarantees an optimal solution and is amenable to computer programming, it also is an impractical method because a large number of time consuming operations must be performed. For example, an attempt to solve the thirteen stop problem from[14] was abandoned when no solution was obtained after one hour was spent on the Control Data G-21 computer.

Dantzig and Ramser[14] described a heuristic procedure for solving the delivery problem when all trucks have the same capacity. After the delivery points are arranged in numerically ascending order with respect to demand and the number of stages required to achieve solution has been calculated, this algorithm synthesizes routes by a stage-wise aggregation of the delivery points, i.e. in stage one, pairs of points are joined, in stage K, groups of K points are joined to other groups of K points. At each stage the points are combined in such a manner that truck capacity is not exceeded and the sum of the interstop distances is minimized.

Although this method is sequential and amenable to computer programming, personal judgement may be required in adjusting the final solution which may not be uniquely defined. Moreover, this procedure places more emphasis on insuring that the trucks are loaded to capacity than on minimizing the total distance traveled. It also appears that this algorithm would require considerable alteration to handle the asymmetric case because direction is not considered when the interstop distance matrix is searched to determine the minimum entry. This method would probably be quite time consuming in solving problems which involve more than 20-30 stops. Each entry of the distance table in stage K, where $K = 2,3,\ldots,N$ and N is the number of stages required, contains the shortest route which starts at the depot, visits 2^K points and returns to the depot. Therefore, many traveling-salesman problems must be solved at each stage. For example, if after 2 stages 100 stops are aggregated into 25 groups of 4 stops each, then the distance table for stage 3 requires the solution of 300 traveling-salesman problems of 9 stops.

Clarke and Wright[9] described a heuristic procedure for solving the delivery problem when all vehicles do not have the same capacity. After the demand points are arranged in numerically ascending order with respect to distance from the origin, the algorithm assigns each stop to one vehicle or route. Then each pair of points is examined to determine the savings which would result if they were linked together on the same route instead of being assigned to different routes. A savings calculation is made only for those points, currently linked to the origin, which could be assigned to the same route without violating vehicle capacity constraints. The two points associated with the maximum savings are then assigned to the same route. After repeating the procedure until no further savings can be calculated, the resulting set of routes is the solution to the problem.

This method is amenable to computer programming and appears to be efficient for solving the delivery problems in the literature. However, it cannot be readily extended to handle the case of the asymmetric interstop distance matrix because direction is not considered in calculating the savings or the criterion for accepting or rejecting the linkage of two points on the same route. Moreover, the largest problem for which solution success with this method has been reported is one involving thirty-two stops[18].

Cochran[10,36] made two modifications to the algorithm of Clarke and Wright[9]. The first change permits the reassignment of vehicles to individual routes each time a vehicle becomes available as a result of the linkage of two points on the same route. The second change allows an upper bound to be placed upon the length of any route. Although the first modification insures that the vehicles will be more fully utilized and the second modification is a useful one, neither the probability of reaching optimality nor the efficiency of the algorithm is increased by their inclusion.

Braun[8] described a simulation approach to the delivery problem. First, delivery points are randomly assigned to an individual route subject to the vehicle capacity constraint. Each route is then improved by applying a traveling-salesman algorithm. After a specified number of sets of routes are developed in this manner, the set which covers the least number of miles is selected as the solution.

Although this procedure is simple and amenable to computer programming, it proved to be less efficient than other available methods with respect to the quality of the routes produced and the amount of computer time required to achieve solution. Moreover, the quality of the sets of routes tends to become poorer as the number of delivery points increases.

After abandoning the branch and bound method for solving the delivery problem, Hayes[21] developed a computerized version of a procedure which an experienced dispatcher might employ for routing trucks. Since this method assumes that the warehouse or depot is located near the center of the scatter of customers and performs poorly when this assumption is not satisfied, it does not appear to be superior to the available heuristic methods.

Gaskell[18] reported on his experiments with the method of Clarke and Wright[9]. He devised five functions for calculating the savings which would result from linking two points on the same route. After applying each of them to a set of delivery problems, he concluded that none was uniformly better and that the function used by Clarke and Wright to calculate savings was a reasonable one.

Although more work has been done on the delivery problem, successful solution has been reported only for relatively small problems with no indication of apparent near future breakthroughs for larger problems involving asymmetric interstop time/distance matrices, routes whose origin and terminus do not coincide, and restrictions upon the length of a route, vehicle capacity, and the number of routes used to service all the stops. Indeed, solution success on the school bus routing problem may provide a means of solving the delivery problem.

III. PROBLEM FORMULATION

A. Statement

Thus far, the school bus routing problem has been solved for an environment which has been greatly simplified by the imposition of various assumptions. Most solutions not only exclude constraints on some of the route characteristics but also assume that the origin and terminus of a bus route coincide, whereas in reality they are not necessarily the same. In fact, there often may be several possible origins for the set of bus routes servicing a particular school. Therefore, it would be desirable to be able to judiciously select the proper combination of origins from the set of possible origins for the system of bus routes servicing a particular school.

With these thoughts in mind, this dissertation is concerned with the determination of all bus routes for a school district. Buses are routed from school to school picking up students as they travel. A heuristic procedure has been developed and programmed in FORTRAN IV to generate efficient bus routes when the school bus routing problem has the following definition:

Given:

1. the number of time periods used by the school district for bussing
2. for each time period
    a. the number and location of schools to be serviced
    b. the number of possible origins for the bus routes
    c. the identification of each origin
    d. the number of buses available at each origin at the beginning of the period

3. for each school to be serviced during the same time period
   a. the identification of the school,
   b. the identification of each stop to be visited,
   c. the number of students assigned to each stop,
   d. the matrix of interstop travel times between each possible origin and every stop assigned to the school,
   e. the matrix of interstop travel times for all pairs of stops except those links involving an origin,
   f. the bus capacity,
   g. the maximum allowable student riding time, and
   h. the criterion for accepting a set of feasible routes as the quasi-optimal solution.

Determine for each school:

1. the set of bus routes and schedules required to provide transportation for all students either to school or from school and
2. a lower bound on the total number of time units required to traverse all the bus routes

such that:

1. no more than the absolute minimum number of routes required to transport the students plus one route will be used for any school,
2. the bus capacity and maximum allowable student riding time constraints will be satisfied,
3. the criterion for accepting a set of feasible routes as

the quasi-optimal solution will be satisfied, and

4. optimization will be with respect to minimizing the total traveling time for a set of routes.

It is assumed that:

1. all buses assigned to a particular school have the same capacity,
2. all routes for a particular school will be subject to the same maximum allowable student riding time constraint,
3. the matrix of interstop travel times may be asymmetric,
4. a stop is assigned to only one route,
5. the traveling time from any bus stop to the terminus is less than the maximum allowable student riding time,
6. the origin and terminus of any bus route do not necessarily coincide,
7. a bus services only one route for a particular school, and
8. the number of buses available at the beginning of period i is adequate to meet the needs of all students to be serviced during period i, i.e. it is possible to service all the routes simultaneously.

B. Discussion of the Problem Statement

Most districts set the opening and closing times of the schools serviced by the same bus fleet so that all students can be transported during two or three non-overlapping bussing periods whose total elapsed time is approximately two and one-half hours. Schools maintaining common hours of operation are serviced during the same bussing period.

The data required to design the routes for each school can be developed either manually or by computer from the map of the area and the associated census tract. Each stop serviced by the school is usually identified by both name and number in order to facilitate communication between the administrators and the bus drivers. The elements of the interstop travel time matrix are calculated from a map of the area so that they include the effects of both distance and expected driving conditions in traveling between every pair of points. Often in an effort to reduce the size of the interstop travel time matrix, bus stops adjacent to each other on the same side of the road are combined into a single stop. The information required to assign students to a bus stop can be extracted from the census tract. Usually upper limits are imposed upon the number of students assigned to a bus stop in order to reduce the noise level and the possibility of landscape damage in residential areas.

Optimization is with respect to not only minimizing the total time required to traverse a set of routes but also with respect to minimizing the number of routes required to transport all the students of a particular school. Both of these factors contribute heavily toward the operational costs of maintaining a bus fleet. Minimizing the number of routes insures that the buses are being utilized to full capacity and also tends to reduce the total traveling time associated with a set of routes by eliminating some of the links between an origin and the first stop of any route and the links between the last stop of any route and the terminus. Since schools usually place more emphasis on minimizing the number of routes required to transport all the students than on minimizing the total traveling time for a set of routes, the maximum allowable number of routes for a school will be considered to be a constraint.

The minimum number of routes required to service all the stops assigned to a school is the smallest integer which is greater than or equal to the quotient of the total number of students to be transported and the bus capacity. Since a riding time constraint is imposed on each route and since all the students assigned to a particular bus stop have to be picked up by the same bus in order to avoid confusion, it may be impossible to service all the bus stops by the minimum number of routes. Therefore, a set of routes is considered feasible if each route satisfies the bus capacity and student riding time constraints and if the set contains at most one more than the minimum number of routes required to transport all the students. Thus, a feasible set of routes contains either the minimum number of routes or one more than the minimum number of routes.

Because the set of bus routes for any school is designed by a heuristic procedure, some method must be devised to determine when a set of routes, acceptable to the school, has been developed. One possible criterion is to accept the best set of feasible routes available after the procedure has been executed a specified number of times. A second possible criterion is to accept the first set of feasible routes developed whose total travel time is less than the product of a given factor, greater than one, and the lower bound upon the total time required to traverse all the routes for a school. Another alternative is to accept as the quasi-optimal solution the best set of feasible routes available at the time at which either of the criteria is first satisfied.

The lower bound upon the total time required to traverse a set of routes for a school is the minimum length of time in which all the bus stops could be serviced by the number of routes included in the set.

In calculating the lower bound, it is assumed that all connections between the origin and the first stop of a route, all the connections between the bus stops, and all the connections between the last stop of a route and the terminus are made in the least time consuming manner. Since the lower bound does not include the effects of the bus capacity and passenger riding time constraints, the probability of developing a set of feasible routes whose total traveling time equals the lower bound is very low. However, even though the lower bound may be unattainable, it provides some measure for assessing the quality of the set of routes developed.

An examination of the set of assumptions under which the problem is to be solved reveals that the environment has not been oversimplified. The bus capacity is based upon the expected girth of children at various age levels, the size of the bus used, and the degree of bus utilization required by the school. The maximum student riding time is dependent upon the size of the area, the scatter of the bus stops and the local or state laws regulating student riding time. Usually, districts consider the bus capacity and the maximum riding time constraints to be fixed for a particular school.

The interstop travel time matrix is characteristically asymmetric for reasons such as: restrictions imposed upon the crossing of busy streets by children, one-way streets, limited access highways, and restrictions imposed on vehicle turns at intersections. Moreover, most schools maintain a bus loading policy which requires that all students assigned to a stop by picked up by the same bus in order to avoid confusion. This restriction implies that a stop is assigned to only one route.

The traveling time from any bus stop to the terminus has to be less than the maximum allowable student riding time in order to insure that no bus stop b- isolated. The assumptions that a bus services only one route for a school and that it is possible to service all the routes simultaneously are imposed in order to facilitate the computational procedure used to develop all the bus routes for a district and do not simplify the environment. Thus, this formulation of the school bus routing problem is considered to be a realistic and a reasonable one.

C. Mathematical Model

The bus routing problem for any school, as previously defined, can be expressed as a zero-one integer programming problem.

Let i = origin of any link

j = terminus of any link

k = number of any possible bus origin

m = route number

K = number of possible different bus origins for a set of routes

M = number of routes allowed for the school

N = total number of bus stops (includes the origin and the terminus) assigned to the school

stop 1 (k) = bus origin number k

stop N = terminus of any route or the school

$t(i,j)$ = number of time units required to travel between bus stop i and bus stop j

$L(j)$ = number of students assigned to bus stop j

$B(k)$ = number of buses available at origin k

C = bus capacity

R = maximum allowable student riding time

$$x(i,j,m) = \begin{cases} 1, & \text{if link } (i,j) \text{ is assigned to route } m \\ 0, & \text{otherwise} \end{cases}$$

The subscript denoting the particular school has been omitted in order to simplify the nomenclature. The words bus and route are used interchangeably.

The problem can be stated as follows:

Find variables $x(i,j,m)$ for all combinations of i, j, k and m where

$i = 1(k), 2, \ldots, (N-1)$

$j = 2, 3, \ldots, N$

$k = 1, 2, \ldots, K$

$m = 1, 2, \ldots, M$

such that the objective function

$$\sum_{j=2}^{N} \sum_{k=1}^{K} \sum_{m=1}^{M} [t(1(k),j)][x(1(k),j,m)]$$

$$+ \sum_{i=2}^{(N-1)} \sum_{j=2}^{N} \sum_{m=1}^{M} [t(i,j)][x(i,j,m)]$$

is minimized while restrictions one through eight, stated below, are satisfied.

Optimization, as previously stated, is with respect to not only minimizing the total time required to traverse a set of routes but also with respect to minimizing the number of routes required to transport all the students of the school. Since it is difficult to work with two objective functions, and since most schools place greater emphasis on the minimization of the number of routes than on the minimization of the total traveling time, the number of routes required to transport all the students of a school will be considered a constraint and the objective function will involve the total traveling time only. The first term of the objective function is the total time spent in traveling from the origin to the first stop of any route; the second term is the total time spent in traveling between the other pairs of points.

The first constraint,

$$\sum_{k=1}^{K} \sum_{m=1}^{M} x(1(k),j,m) + \sum_{i=2}^{(N-1)} \sum_{m=1}^{M} x(i,j,m) = 1, \quad \text{for } j = 2,3,\ldots,(N-1),$$

insures that any bus stop j, where j is not the origin or terminus of any route, will be the terminus of exactly one link on one route. There will be (N-2) constraints of this type.

The second constraint,

$$\sum_{j=2}^{N} \sum_{m=1}^{M} x(i,j,m) = 1, \quad \text{for } i = 2,3,\ldots,(N-1),$$

insures that bus stop i, where i is not the origin or terminus of any route, will be the origin of exactly one link on one route. There will be (N-2) constraints of this type.

The third constraint,

$$\sum_{j=2}^{(N-1)} \sum_{k=1}^{K} \sum_{m=1}^{M} x(1(k),j,m) = M,$$

insures that only M routes are used by counting the number of links between the origin of each route and the first stop serviced by the route.

The fourth constraint,

$$\sum_{i=2}^{(N-1)} \sum_{m=1}^{M} x(i,N,m) = M,$$

also insures that only M routes are used by counting the number of links between the last stop serviced by a route and the terminus or school.

The fifth constraint,

$$\sum_{j=2}^{(N-1)} \sum_{m=1}^{M} x(1(k),j,m) \le B(k), \quad \text{for } k = 1,2,\ldots,K,$$

insures that the number of routes starting at a particular origin does not exceed the number of buses available at that origin for the school. There will be K constraints of this type.

The sixth constraint,

$$\sum_{j=2}^{(N-1)} \sum_{k=1}^{K} [x(1(k),j,m)][L(j)] + \sum_{i=2}^{(N-1)} \sum_{j=2}^{(N-1)} [x(i,j,m)][L(j)] \le C,$$

for m = 1,2,...,M, insures that the bus capacity will not be exceeded by any route. The first term of the constraint counts the number of students picked up at the first bus stop serviced by the route and the second term counts the students picked up at the other stops assigned to the route. There will be M constraints of this type.

The seventh constraint,

$$\sum_{i=2}^{(N-1)} \sum_{j=2}^{N} [t(i,j)][x(i,j,m)] \le R, \quad \text{for } m = 1,2,\ldots,M,$$

insures that no route will exceed the maximum allowable student riding time where the riding time is counted from the first pick-up point. There will be M constraints of this type.

The eighth constraint,

$$\sum_{p=1}^{(r-1)} x(i_p,i_{p+1},m) + x(i_r,i_1,m) \le (r-1), \quad \text{for } m = 1,2,\ldots,M,$$

where $i_1, i_2, \ldots, i_r$ ranges over all permutations of subsets of the bus stops $\{2,3,\ldots,(N-1)\}$ of size r, and $2 \le r \le (N-1)$, prevents the formation of a loop, a route which starts and ends at the same point. There will be $M \sum_{k=2}^{(N-2)} P_k^{(N-2)}$ constraints of this type where P_x^z denotes the permutation of z things taken x at a time. Since $\sum_{k=2}^{(N-2)} P_k^{(N-2)}$ is greater than $2(N-2)!$, there will be more than $2M(N-2)!$ constraints of this type. None of the permutations involve the origin or terminus because no link is allowed to start at the terminus and no link is allowed to end at the origin in the case of the general route whose origin and terminus do not coincide. This is accomplished by setting the travel times associated with these links equal to infinity, i.e. the elements of column one and row N of the interstop travel time matrix are assigned large values. To illustrate the manner in which the eighth constraint prevents the formation of a loop, let $r = 2$, $m = 4$, $i_1 = 3$, $i_2 = i_r = 5$, $x(3,5,4) = 1$, $x(5,3,4) = 1$ and the rest of the variables, $x(i,j,4)$ equal zero. In this example, route four starts at stop three, proceeds to stop five, and then returns to stop three. According to the eighth constraint, $x(3,5,4) + x(5,3,4) = 2 = r$ and route four is a loop.

This zero-one integer programming model would have to be solved twice. First, it would be solved for M equal to the minimum number of routes required to transport all the students of the school and then for M equal to one route plus the minimum number of routes required by the school.

Since i may assume any of (K+N-2) values, j any of (N-1) values and m any of M values, there are (K+N-2)(N-1)(M) variables, x(i,j,m), in this problem. Moreover, there are more than 2M(N-2)! constraints. A normal bus routing problem for one school involving ten origins, twenty routes and one hundred bus stops would require 213840 variables and more than 40(98)! constraints. Since the available integer programming algorithms are clearly unable to handle problems of this magnitude, a heuristic procedure is the only recourse for developing all the bus routes for a school district.

IV. METHOD OF SOLUTION

A. General Procedure

Bus routing for a school district is accomplished by suboptimization from period to period. Bus routing for a period is accomplished by suboptimization over the sets of routes developed for each of the schools serviced during the period. Quasi-optimal bus routing for an individual school is accomplished by an algorithm based selective enumeration procedure. A detailed description of each step of this enumeration procedure is given in the following sections. It can be summarized as follows:

1. Assuming that the maximum allowable number of routes will be used by all schools, determine for every school the number of bus routes which should start from each origin supplying buses for the period during which the school is being serviced so that the total estimated traveling time required by all the routes for the period is minimized.
2. For each school, select the location which serves as the origin for the greatest number of routes as determined in step one. This point will be called the "super-origin" for the school.
3. Determine by either using the Nearest City Approach or Algorithm A, a trial route which starts at the super-origin, visits every stop once and terminates at the school. The method used to develop this trial route is dependent upon the number of times step three has been executed.
4. Partition the single route determined in step three into

individual routes which satisfy bus capacity and passenger riding time constraints. The order of the stops determined in step three is preserved during the partitioning process.

5. If the set of routes contains no more than the maximum allowable number of routes, then proceed to step six. Otherwise, return to step three to generate a different trial route.

6. Improve each of the individual routes by Algorithm A, a modified traveling-salesman type algorithm. This step attempts to reduce the traveling time required by the individual route while preserving the assignment of stops to the route made by the partitioning procedure.

7. Determine the current best set of routes developed by the procedure.

8. If the set of routes obtained in step seven satisfies the acceptance criterion specified by the school, then proceed to step nine. Otherwise, repeat steps three through eight until either the acceptance criterion is satisfied or the algorithms used to generate the trial route in step three are exhausted, i.e. they are unable to generate another different trial route.

9. If the results of step one specify that all the routes for the school should start at the super-origin, then this acceptable set of routes is considered to be the quasi-optimum solution. Otherwise proceed to step ten.

10. Allocate the remaining origins specified by step one in such a way that the additional number of time units traveled

will be minimum. Then, this modified set of acceptable routes is considered to be the quasi-optimum set of routes for the school.

After bus routing has been completed, the schedule or timetable for each individual route is calculated. Each schedule gives the bus load and the arrival time at each stop serviced by the route. The time required to load the bus at each stop and to unload the bus at the school is not included in calculating the timetable. In addition, a lower bound upon the total number of time units required to traverse all the routes required by the school is calculated.

B. Selection of Bus Route Origins

At the beginning of the first bussing period all buses are located at the garage(s) maintained by the school district. For all other bussing periods the available buses are initially located at the schools serviced during the previous period. It is assumed that the number of buses available at the beginning of any bussing period is adequate to meet the needs of all students to be transported during the period, i.e. it is possible to service all the routes for the period simultaneously. For each school j serviced during the same period, the determination of the number of bus routes which should start at each origin i can be expressed as a transportation problem.

Let $n1$ = number of origins supplying buses for the period

$n2$ = number of schools serviced during the period

$x(i,j)$ = number of routes starting at origin i and ending at school j

$c(i,j)$ = average estimated time required to traverse any route starting at origin i and ending at school j

A(i) = number of buses available at origin i at the beginning of the period

R(j) = number of routes required by school j

B(j) = bus capacity specified by school j

The problem can be stated as follows:

Find variables x(i,j) for all combinations of i and j, where i = 1,2,...,n1 and j = 1,2,...,n2, such that the objective function

$$\sum_{i=1}^{n1} \sum_{j=1}^{n2} [c(i,j)] [x(i,j)]$$

is minimized and restrictions one through four, stated below, are satisfied. Optimization is with respect to minimizing the total estimated traveling time required by the n2 sets of routes associated with the period.

The first constraint,

$$\sum_{i=1}^{n1} x(i,j) = R(j), \quad \text{for } j = 1,2,\ldots,n2,$$

insures that all the routes required by school j are assigned to an origin. There will be n2 constraints of this type.

The second constraint,

$$\sum_{j=1}^{n2} x(i,j) = A(i), \quad \text{for } i = 1,2,\ldots,n1,$$

insures that that number of bus routes starting at origin i equals the number of buses available at origin i. It is assumed that a bus services only one route for a school. There will be n1 constraints of this type.

The third constraint,

$x(i,j) \geq 0$, an integer, for all i and j,

is self-explanatory.

The fourth constraint,

$$\sum_{i=1}^{n1} A(i) = \sum_{j=1}^{n2} R(j)$$

insures that a feasible solution exists. It can be verified by observation. Since the first constraint implies $\sum_{j=1}^{n2} \sum_{i=1}^{n1} x(i,j) = \sum_{j=1}^{n2} R(j)$ and the second constraint implies $\sum_{j=1}^{n2} \sum_{i=1}^{n1} x(i,j) = \sum_{i=1}^{n1} A(i)$, then indeed the fourth constraint must be satisfied.

Because it is assumed that $\sum_{i=1}^{n1} A(i) \geq \sum_{j=1}^{n2} R(j)$ at the beginning of any bussing period, a fictitious school may have to be introduced to use the extra available buses in order to satisfy the fourth constraint.

The parameters n1, n2, A(i), and B(j) are given data for the problem. However, the parameters R(j) and c(i,j) for i = 1,2,...,n1 and j = 1,2,...,n2 must be calculated. The R(j) are determined as follows:

Let L(j,k) = number of students assigned to stop k associated with school j

n(j) = number of stops assigned to school j (including the origin and terminus)

$m(j)$ = minimum number of routes required to transport all students of school j

The values of $L(j,k)$ and $n(j)$ where $j = 1,2,\ldots,n2$ and $k = 1,2,\ldots,n(j)$ are given data for the problem.

$$m(j) = \left[\sum_{k=1}^{n(j)} L(j,k) \Big/ B(j) \right] + 0.999999 \text{ truncated to the}$$

nearest integer.

$R(j) = m(j) + 1, \qquad \text{for } j = 1,2,\ldots,n2$

The parameter $R(j)$ is used in the determination of the number of routes which should start at origin i for school j instead of $m(j)$ because the bus capacity and student riding time constraints often make the generation of a set of feasible routes containing only $m(j)$ routes unattainable.

The $c(i,j)$ are calculated as follows:

Let $LB(i,j)$ = lower bound on the total traveling time required by $R(j)$ routes, all of which start at origin i and end at school j

$c(i,j) = LB(i,j) \,/\, R(j) \qquad \text{for } i = 1,2,\ldots,n1 \text{ and } j = 1,2,\ldots n2$

The $LB(i,j)$ are calculated under the assumption that all the connections between origin i and the first stop of each of the $R(j)$ routes, all the connections between the pairs of bus stops, and all the connections between the last stop of each of the $R(j)$ routes and the terminus are made in the least time consuming manner. A computational procedure for determining the lower bound upon the total traveling time of a set of routes is described in section V-B.

The $c(i,j)$ are based upon a lower bound instead of an upper bound because an attempt is being made to develop a set of routes whose total traveling time approaches this lower bound.

$c(i,j) = 0$ when school j is a fictitious school.

The origin associated with the maximum $x(i,j)$ for school j is defined to be the super-origin for school j. In case of a tie, the super-origin is selected arbitrarily.

The allocation of a number of routes for school j to origin i and the selection of a super-origin for each school serviced during the same period is determined once.

C. Determination of Trial Routes: Nearest-City Approach

A trial route which starts at the super-origin, visits every stop once, and terminates at the school, the route that an infinite capacity bus would traverse, is determined either by the Nearest City Approach or by Algorithm A discussed in the next section. The Nearest City Approach can be described as follows:

Let n = number of stops assigned to the school (including origin and terminus)

stop 1 = origin of route

stop n = terminus or school

r = number of the trial route being generated

where $r = 1,2,\ldots,(n-2)$.

The subscript denoting the particular school has been omitted in order to simplify the nomenclature.

This trial route generated is one in which an "infinite capacity" bus starts at the super-origin, proceeds to stop $(r+1)$ and then repeatedly selects as its next stop that point which is nearest to its present stop and which has not yet been serviced until it reaches the terminus or school. The current best trial route, the one requiring the least traveling time, is determined and saved each time that this process is executed. Since this procedure specifies the first bus stop visited, a different trial route is generated each time. If the acceptance criterion has not been satisfied after (n-2) trial routes have been generated, then Algorithm A is used to develop succeeding trial routes from the best trial route previously determined.

D. Determination of Trial Routes: Algorithm A

Algorithm A, a systematic procedure for decreasing the total time required to traverse the infinite capacity bus route, is an extension of Algorithm 1 developed by the author[29,30]. After the Nearest City Approach has been exhausted, the trial routes are generated by applying Algorithm A to the best infinite capacity bus route available. Initially Algorithm A is applied to the best route determined by the Nearest City Approach. Thereafter, Algorithm A is applied to the most recent trial route it generated.

This algorithm determines sets of three links which can be changed simultaneously without destroying the continuity of the tour, the non-coincidence of the origin and the terminus of the route, and the direction of the unchanged portions of the route. The latter constraint is necessary, because the procedure is applicable to non-symmetric as well as symmetric problems. If the time required to traverse the three new links is less than the time required to traverse the links which they replace, then the new route becomes the next trial route and the best infinite capacity bus route available. If the proposed change results in no improvement, then another set of changes is determined and examined. Algorithm A is repeated until it is unable to improve the best trial route available, i.e. Algorithm A is exhausted.

The sets of three link changes in the current tour through a network of n points or stops in which the origin and terminus do not coincide are generated as follows for all combinations of i and j where $1 \leq i \leq (n-1)$, $2 \leq j \leq (n-1)$, stop 1 is the super-origin and stop n is the terminus or school:

1. New link 1 starts at point i and ends at point j where $i \neq j$ and point $j \neq$ the point which follows point i in the current tour.
2. New link 2 starts at point k and ends at the point which follows point i in the current route where $k \neq n$.

   k is cycled as follows:

point k_1	=	point j
point k_2	=	point which follows point j in the current route
⋮		
point k_L	=	point which lies (L-1) consecutive positions after point j in the current route in a clockwise direction
⋮		
point k_m	=	point which precedes point i in the current route

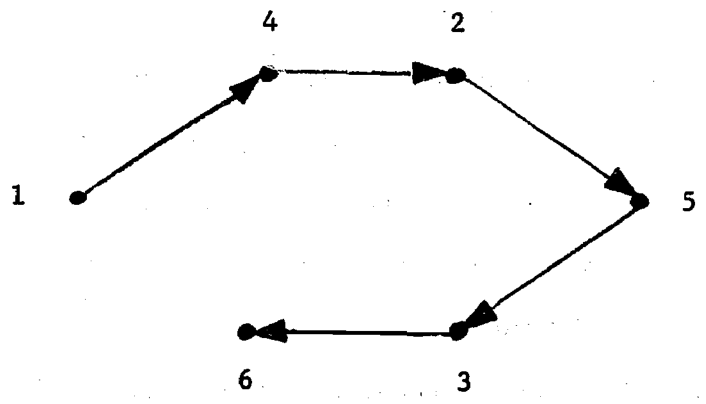
   In order to determine the complete range of values for index k associated with a particular i,j combination, it is necessary to assume the existence of a fictitious link between the terminus and origin of the current route while counting the consecutive positions after point j in a clockwise direction.

3. New link 3 starts at the point which precedes point j in the current route and ends at the point which follows point k in the current route.

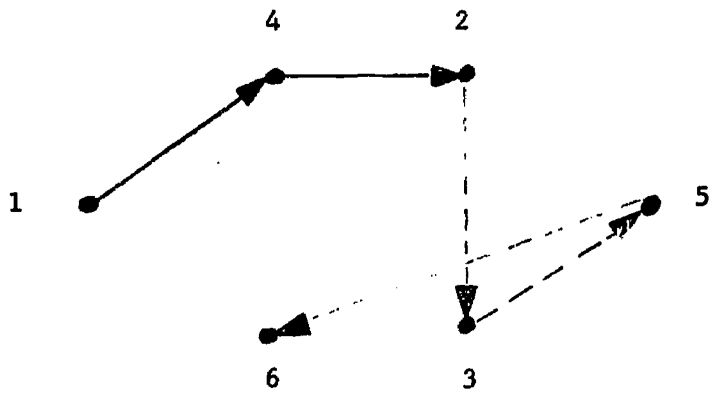
For illustrative purposes, Algorithm A will be applied to a route containing six points for one combination of i and j and the entire range of index k associated with it. Point one is the super-origin and point six is the terminus or school. The number associated with each node is the permanent identification number of the stop and corresponds to its position in the interstop travel time matrix, eg. stop four data would form row four of the interstop travel time matrix.

Let $i = 2$ and $j = 3$ be the i,j combination

R_0 = 1-4-2-5-3-6 be the current route

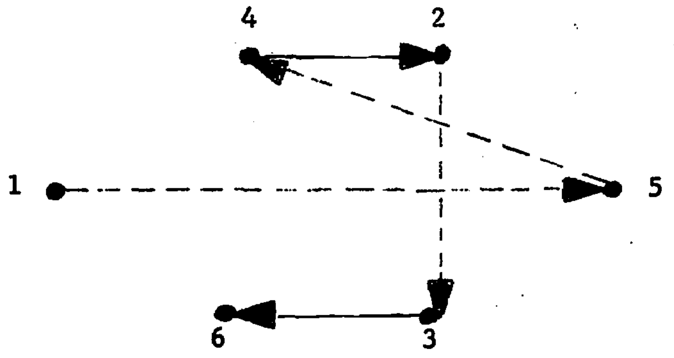


The first value assumed by k is the number of point j or three. The proposed route is R_1 = 1-4-2-3-5-6. Links 2-5, 5-3, and 3-6 are replaced by new links 2-3, 3-5, and 5-6, indicated by the dashed arcs.

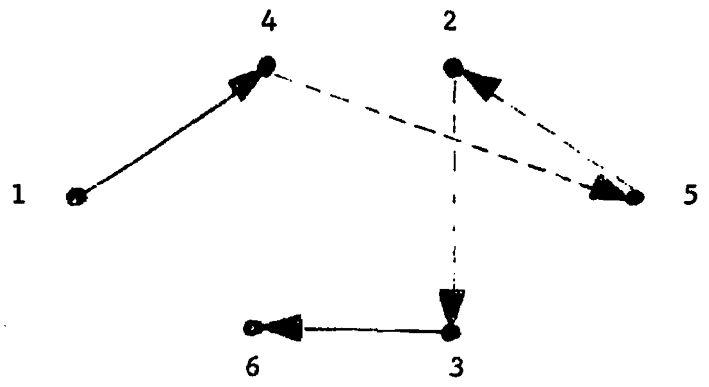


The second value assumed by k is the number of the point which follows point j or six. However, no new route can be proposed by Algorithm A for this combination of i,j, and k because the value assigned to index k is the number associated with the terminus of the route. Since k has not yet assumed the value of the identification number of the point preceding point i in the current route, four, the cycle for index k is incomplete.

The third value assumed by k is the number of the point which lies two consecutive positions after point j, in a clockwise direction, or one. The proposed route is R_2 = 1-5-4-2-3-6. Links 1-4, 2-5, and 5-3 are replaced by new links 1-5, 5-4, and 2-3.



The fourth value of k is the number of the point which lies three consecutive positions, in a clockwise direction, after point j in the current route or four. This is the last k which can be generated for the combination i = 2 and j = 3 because the point numbered four precedes point i in the current route. The proposed route is R_3 = 1-4-5-2-3-6. Links 4-2, 2-5, and 5-3 are replaced by new links 4-5, 5-2, and 2-3.



When Algorithm A cannot improve the best trial route available, it is said to be exhausted. This best trial route which Algorithm A cannot improve is considered to be the quasi-optimal modified traveling-salesman route and the last possible trial route. A computational procedure for Algorithm A is described in section V-A.

E. Partitioning Procedure

The partitioning procedure is applied to every trial infinite capacity bus route determined by either the Nearest City Approach or Algorithm A. This procedure requires the following additional information:

1. a student load vector specifying the number of students

assigned to each stop,

2. the bus capacity, and

3. the maximum allowable riding time of the students picked up at the first stop of any route.

The partitioning procedure generates a set of bus routes each of which starts at the super-origin, visits the stops of the trial infinite capacity bus route in the order previously determined until the bus is loaded properly and proceeds to the terminus or school. At each stop, the bus load count is incremented by the appropriate element of the student load vector and the time tally is incremented by the traveling time from the previous stop. When either the bus capacity or the riding time constraint is about to be exceeded, the previous stop becomes the last one serviced by the bus before proceeding to the school. The next route starts at the super-origin and proceeds directly to that stop of the trial infinite capacity bus route which immediately follows the last stop serviced. All individual bus routes are determined in the same manner. The sequence of stops generated by either the Nearest City Approach or by Algorithm A is preserved throughout this procedure. If the set of routes determined by the partitioning procedure contains no more than the maximum allowable number of routes, then the improvement process, described in the next section, is applied to each route of the set. Otherwise, another trial infinite capacity bus route is generated by either the Nearest City Approach or Algorithm A.

F. Improvement Process

Each individual route belonging to a feasible set of routes developed by the partitioning procedure is then improved by application of Algorithm A, previously described in section IV-D.

First, form a submatrix of the given interstop traveling time matrix consisting of those elements associated with the super-origin, the stops serviced by the individual route, and the terminus or school. The infinite capacity bus route corresponding to this submatrix is the individual route being improved. Then Algorithm A is applied to this route until it can make no further improvement. This two step procedure is repeated until all the individual bus routes of the feasible set have been improved. The given matrix of interstop travel times between each possible origin and every stop assigned to the school and the given matrix of interstop travel times for all pairs of stops except those links involving an origin are preserved at all times.

G. Acceptance Criteria

The best available set of routes, all of which start at the super-origin, for a particular school j will be considered to form the basis of the quasi-optimal solution if one of the following criteria is satisfied:

1. The total time required to traverse this set of routes is less than or equal to the product of a factor specified by the transportation director, greater than one, and the lower bound on the total time required to traverse the number of routes contained in the best available set of routes, assuming that all of them start at the super-origin.
2. The total number of trial routes generated is about to

exceed some number specified by the transportation director. Thus, the school administrator has an opportunity to specify the degree of optimality required for the set of routes accepted as a final solution. In the event that neither of the criteria are satisfied and no further trial infinite capacity bus routes can be developed because both the Nearest City Approach and Algorithm A have been exhausted, then the best available set of routes will become the basis of the quasi-optimal solution by default. The latter situation is one in which the capability of the method is unable to satisfy the requirements of the school.

H. Final Allocation of Origins to Individual Routes

The feasible set of individual bus routes which satisfy the acceptance criterion specified by school j all start at the super-origin. If all the routes for school j should start at the super-origin, as previously determined, then a final allocation of origins to individual routes is unnecessary. However, if all the routes for school j do not start at the super-origin, then the remaining origins are allocated in such a way that the additional number of time units traveled will be minimum. This final allocation problem can also be expressed as a transportation problem.

Let n1 = number of origins supplying buses for the period

n3(j) = number of routes belonging to the feasible set of routes satisfying the acceptance criterion specified by school j

x(i,j) = number of routes which should start at origin i and end at school j

$d(i,j,k)$ = total traveling time of route k assigned to school j when it starts at origin i

$$y(i,j,k) = \begin{cases} 1 & \text{when route k assigned to school j starts at origin i} \\ 0 & \text{otherwise} \end{cases}$$

The problem can be stated as follows:

Find variables $y(i,j,k)$ for a fixed j and all combinations of i and k, where $i = 1,2,\ldots,n1$ and $k = 1,2,\ldots,n3(j)$, such that the objective function

$$\sum_{i=1}^{n1} \sum_{k=1}^{n3(j)} [d(i,j,k)]\,[y(i,j,k)]$$

is minimized and restrictions one through four, stated below, are satisfied. Optimization is with respect to minimizing the total traveling time required by the n3(j) routes accepted by school j.

The first constraint,

$$\sum_{i=1}^{n1} y(i,j,k) = 1, \qquad \text{for } k = 1,2,\ldots,n3(j),$$

insures that a route is assigned to only one origin. There will be n3(j) constraints of this type.

The second constraint,

$$\sum_{k=1}^{n3(j)} y(i,j,k) = x(i,j), \qquad \text{for } i = 1,2,\ldots,n1$$

insures that the number of bus routes starting at origin i equals the number of buses previously assigned to school j from origin i. There will be n1 constraints of this type.

The third constraint,

$$y(i,j,k) = 0 \text{ or } 1, \qquad \text{for all } i \text{ and } k,$$

is self-explanatory.

The fourth constraint,

$$\sum_{i=1}^{n1} x(i,j) = n3(j)$$

insures that a feasible solution exists. Since the first constraint implies $\sum_{k=1}^{n3(j)} \sum_{i=1}^{n1} y(i,j,k) = \sum_{k=1}^{n3(j)} 1 = n3(j)$ and the second constraint implies $\sum_{i=1}^{n1} \sum_{k=1}^{n3(j)} y(i,j,k) = \sum_{i=1}^{n1} x(i,j)$, then indeed the fourth constraint must be satisfied. Since $\sum_{i=1}^{n1} x(i,j)$ equals the maximum allowable number of routes and $n3(j) \le \sum_{i=1}^{n1} x(i,j)$, a fictitious route may have to be introduced to use the extra available bus in order to satisfy the fourth constraint.

The parameter n1 was given; the parameter n3(j) was determined by the partitioning procedure described in section IV-E. The values of $x(i,j)$, the number of bus routes which should start at origin i and end at school j, were determined by solving the transportation problem described in section IV-B. The $d(i,j,k)$ are calculated as follows:

Let $t(s,j,k)$ = total time required to traverse route k when it starts at the super-origin and ends at school j

$l(s,j,k)$ = traveling time between the super-origin and the first stop serviced by route k associated with school j

$l(i,j,k)$ = traveling time between origin i and the first stop serviced by route k associated with school j

$d(i,j,k) = t(s,j,k) - l(s,j,k) + l(i,j,k)$

$d(i,j,k) = 0$ when k is a fictitious route.

After the final allocation of origins to the individual bus routes for school j has been completed, a lower bound upon the total traveling time of the number of routes required by school j is calculated. This procedure is described in section V-B.

V. COMPUTATIONAL PROCEDURE

A. Algorithm A

A computational scheme for Algorithm A is described by the flow chart given in Figure 1. The nomenclature used in the flow chart is:

N	=	total number of bus stops (including the origin and the terminus)
[M]	=	a square matrix of order (N+1) composed of an NxN interstop travel time matrix augmented by an additional row and an additional column
M(I,J)	=	number of time units required to travel from bus stop I to bus stop J: $1 \leq I \leq N$, $1 \leq J \leq N$, $I \neq J$
M(I,I)	=	number of time units required to travel from bus stop I to the stop which immediately follows it in the current modified traveling-salesman route
M(I,N+1)	=	identification number of the bus stop which immediately follows bus stop I in the current modified traveling-salesman route
M(N+1,J)	=	identification number of the bus stop which immediately precedes bus stop J in the current modified traveling-salesman route
M(N+1,N+1)		is not used
01	=	origin of new link 1
T1	=	terminus of new link 1
02	=	origin of new link 2
T2	=	terminus of new link 2
03	=	origin of new link 3

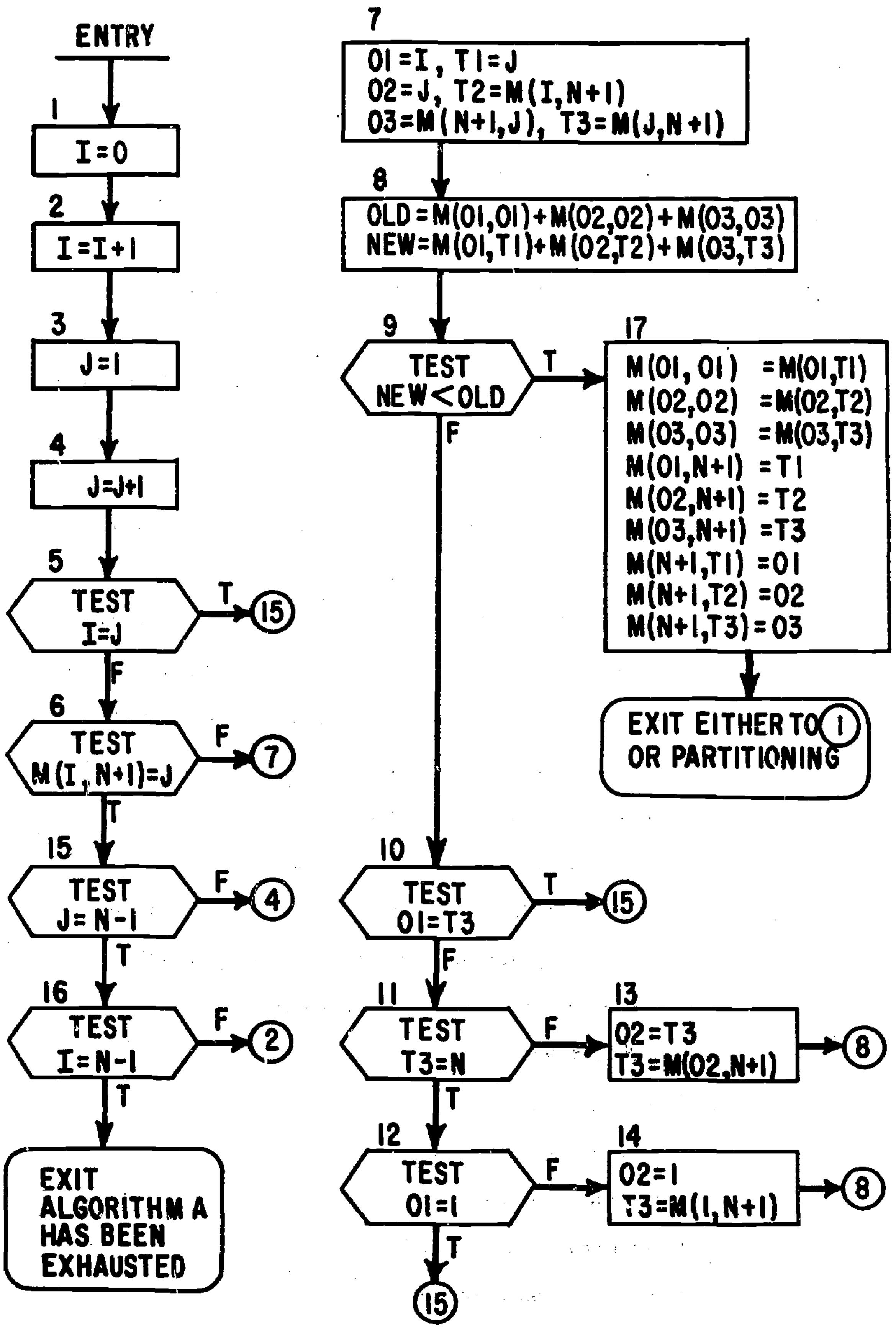


Fig.1 ALGORITHM A

T3 = terminus of new link 3

The first row of [M] is related to the origin and row N is related to the terminus of the modified traveling-salesman route.

Blocks 1-4 provide for the initialization and incrementation of index I and index J.

Blocks 5-6 prevent the creation of an illegal combination of I and J. Block 5 prevents a change in which indexes I and J are identical. Block 6 prevents a change in which stop J immediately follows stop I in the current tour and thereby eliminates the possibility of generating a new route that is identical to the old one.

Block 7 initializes the origin and terminus of each of the three new links associated with the current I,J combination.

Block 8 calculates the time required to traverse the three old links which are candidates for replacement and the time required to traverse the proposed new links.

Block 9 determines whether the proposed set of new links reduces total transit time.

Block 10 tests whether index k, the origin of new link 2, has assumed all possible values for the current combination of indexes I and J.

Block 11 tests whether the next value of index k would be the identification number assigned to the terminus of the current route.

Block 12 determines whether it would be possible to assign another value to index k by introducing a fictitious link between the terminus and origin of the current route.

Block 13 calculates the new value of index k, the origin of new link 2 as described in Section IV-D, and the terminus of new link 3 when index k does not assume the value of the identification number of the terminus of the current route.

Block 14 calculates the new value of index k, the origin of new link 2, and the terminus of new link 3 after a fictitious link was assumed to exist between the origin and terminus of the current route.

Blocks 15-16 determine whether indexes I and J can be further updated. If I cannot be updated, then Algorithm A has been executed for all possible combinations of I and J. If Algorithm A is exhausted while it is being used to generate a trial infinite bus capacity route prior to execution of the partitioning procedure, then the best available set of routes will have to be accepted as the quasi-optimal solution for the school. Exit will be to the final allocation of origins procedure. If Algorithm A is exhausted while it is being used to improve an individual bus route, then exit is to the improvement of the next route of the set.

Block 17 incorporates the three new links, that have been found to reduce total transit time, into the current modified traveling-salesman route by altering row (N+1) and column (N+1) which store the sequence of stops in the new tour and by inserting the new transit times into the main diagonal elements. If Algorithm A produces an advantageous change while it is being used to create a new trial infinite capacity bus route, then exit is to the partitioning procedure. If Algorithm A is being used to improve an individual bus route when an advantageous change occurs, then Algorithm A is restarted and exit is to Block 1.

B. Lower Bound

The lower bound on the total time required to traverse the set of routes accepted as the quasi-optimal solution by the school is calculated under the assumption that all the connections between an origin and the first stop of any route, all the connections between the pairs of bus stops, and all the connections between the last stop of any route and the terminus are made in the least time consuming manner.

Let N = total number of bus stops (including origin and terminus)

M = rectangular matrix with $(N-1)$ rows and N columns

$M(I,J)$ = number of time units required to travel from bus stop I to bus stop J: $2 \le I \le N$, $1 \le J \le N$, $I \neq J$

$N1$ = number of origins supplying buses for the school

$M1$ = rectangular matrix with $N1$ rows and N columns

$M1(K,J)$ = number of time units required to travel from origin K to bus stop J: $1 \le K \le N1$, $1 \le J \le N$

$NR(K)$ = number of routes starting at origin K: $1 \le K \le N1$

$$Q = \sum_{K=1}^{N1} NR(K)$$

$[M]$ is the given matrix of interstop travel times for all pairs of stops except those links involving an origin. Since the origin of a route is not allowed to be the terminus of a link, $M(I,1) = \infty$ for $2 \le I \le N$. Moreover, $M(N,J) = \infty$ for $1 \le J \le N$ because the terminus of a route is not allowed to be the origin of a link. These two restrictions are necessary because the method of solution is applicable to the general routing problem in which the origin and terminus do not coincide.

[M1] is the given matrix of interstop travel times between each origin servicing the period and every stop assigned to the school. Since no link is allowed between the origin and terminus of a route, $M1(K,N) = \infty$ for $1 \leq K \leq N1$. Moreover, $M(I,I) = \infty$ for $2 \leq I \leq N$ and $M1(K,1) = \infty$ for $1 \leq K \leq N1$ because no loop is permitted at any bus stop.

Parameters N1 and NR(K) where $1 \leq K \leq N1$ and $NR(K) \neq 0$ are determined by the final allocation of origins to routes procedure described in Section IV-H.

The lower bound is calculated as follows:

1. Form [M2], a square matrix of order (Q+N-1), by the procedure described below.
2. Reduce [M2] until there is at least one zero in every row and column. This is accomplished by subtracting the smallest element in each row from every element in the row, and then subtracting the smallest element in each column of the remaining matrix from every element in the column. The lower bound on the final set of routes is the total reduction or the sum of the elements subtracted from the rows and columns.

Starting at row one, [M2] is formed as follows:

1. For each K, $1 \leq K \leq N1$, row K of [M1] is stored NR(K) times in [M2], i.e. each route starting at origin K contributes one row to [M2]. This rectangular submatrix consisting of Q rows and N columns occupies the upper left portion of [M2]. During the reduction process, this submatrix provides

that portion of the lower bound contributed by traveling between any origin and the first stop of a route.

2. [M] is stored in rows (Q+1) through (Q+N-1) of [M2]. During the reduction process the first (N-1) columns of this submatrix provide that part of the lower bound contributed by traveling between any pair of points.

3. Steps one and two create columns one through N of [M2]. Column N is then stored (Q-1) more times in [M2], i.e. columns N through (Q+N-1) of [M2] are the same vector whose elements represent the time required to travel between any bus stop and the terminus. During the reduction process, the submatrix occupying rows one through (Q+N-1) and columns N through (Q+N-1) of [M2] provides that portion of the lower bound contributed by traveling between the last stop of any route and the terminus.

When the lower bound has to be calculated for a set of routes, all of which start at the same origin, then N1 = 1 and NR(1) = Q equals the number of routes contained in the set for this procedure.

VI. EVALUATION

Any heuristic procedure should be evaluated on two bases. First, in order to be acceptable, the method must be reasonable with respect to the size of the problem it can handle, the assumptions it imposes and the logical processes it uses. Second, in order to be practical, the procedure must be able to yield answers to the problem being considered at a cost commensurate with the value received.

A. Model and Method

The model developed in this dissertation is a general one which utilizes variables that are applicable to all school systems. Moreover, it requires few assumptions that simplify the real world environment. The method used to obtain a solution for the model reduces the routing of buses for an entire school district to a set of sequential steps that can be readily programmed for a digital computer. Furthermore, this set of sequential steps is arranged into groups called iterations or passes such that a "reasonably good" feasible routing system can be obtained after a few iterations have been completed. Thus, the user of the model and method can specify the degree of optimality desired for the routing system being developed.

The procedure used to develop a set of bus routes for each school is logical. At the beginning of a period, the available buses at the origins are allocated to every school serviced during the period so that the total estimated traveling time required by all the routes developed for the period is minimized. This is accomplished by solving a transportation problem in which the elements of the requirements vector are the maximum allowable number of buses required by a school, the elements of the availability vector are the number of buses located at each origin at the beginning of the period, and the elements of the cost matrix are based upon the lower bound on the total traveling time required by each combination of origin and school. Since the bus capacity and riding time constraints often make the generation of a feasible set of routes containing the absolute minimum number of routes unattainable, the selection of this requirement vector is considered reasonable. Moreover, because an attempt is being made to develop a set of routes whose total traveling time approaches the lower bound, the estimation of the elements of the cost matrix used by the method is a logical choice.

The determination of a trial route which starts at the "super-origin", visits every stop once and terminates at the school by either the Nearest City Approach or Algorithm A tends to group stops that are located in the same neighborhood. Since a school usually services an area of less than twenty square miles, the variability associated with the magnitude of the elements of the interstop travel time matrix is low. Thus, arranging the bus stops to be serviced into groups located in the same neighborhood is logical. The first (n-2) different trial routes, where n is the total number of stops assigned to the school including the origin and the terminus, are developed by the Nearest City Approach because it is a rapid and relatively efficient process. After all trial routes are generated by the Nearest City Approach, the route requiring the least traveling time is saved in order to reduce the number of trial routes which Algorithm A will generate before it is exhausted and to insure that Algorithm A does not create one of the routes previously developed.

Algorithm A is applied to the best available trial route because a good "infinite capacity" bus route usually partitions into a good routing system even though the best "infinite capacity" bus route does not necessarily generate the best set of individual routes. Although Algorithm A is a slightly slower process than the Nearest City Approach, it guarantees that the next trial route developed will require less traveling time. Both of the algorithms were selected because they are rapid, efficient, and exhausted in a finite number of steps.

The partitioning procedure quickly generates routes which not only satisfy the bus capacity and passenger riding time constraints but also service stops in the same general area. If the set of routes developed contains at most one more than the absolute minimum number of routes required by the school, then each route of the set is improved by application of Algorithm A until it is exhausted while preserving the assignment of stops to the route made by the partitioning process. This improvement procedure may reduce the traveling time required by the individual route and thus tends to reduce the total transit time of the routing system.

Minimizing the number of routes included in a routing system insures that the buses are being utilized to full capacity and also tends to reduce the total traveling time required by the set of routes by eliminating some of the links between an origin and the first stop of any route and the links between the last stop of any route and the terminus. Since at most ten students are usually assigned to a bus stop in order to reduce the noise level and the possibility of landscape damage in residential areas, satisfying the maximum allowable number of routes in a routing system constraint presents no problem. However, if a transportation director decides to lump the bus stops extensively so as to reduce the size of the interstop travel time matrix or if the population density is such that many students are assigned to one stop, then determining a feasible routing system containing at most one more than the absolute minimum number of routes may be unattainable. The user of this method must then adjust the set of constraints by either increasing the maximum allowable number of routes in the routing system or by reducing the bus capacity. Another alternative is the addition of bus stops to reduce the number of students assigned to individual stops.

The specification of a criterion for accepting a set of routes gives the transportation director the opportunity to select the degree of optimality under which the routing system will be developed. This acceptance criterion is applied to a routing system when all the routes start at the "super-origin" because allocating an origin to each route every time a feasible routing system is generated would require more computer time than is warranted by the improvement which would be realized.

The final allocation of origins to individual routes, if necessary, is accomplished by solving a transportation problem in which the elements of the availability vector are the number of routes which start at each origin as determined by step one of the solution procedure and the elements of the requirements vector are all equal to one. Each element of the cost matrix is the total traveling time required by a route when it starts at a particular origin. Optimization is with respect to minimizing the total traveling time required by the final set of routes for the school. Initially, the solution procedure develops routing systems in which all the individual routes start at the "super-origin" in order to minimize the effects of this final allocation of origins to routes. After an origin has been assigned to each route of the set, no attempt is made to further reduce the transit time required by each route because the improvement process may create a violation of the student riding time constraint.

Although the final lower bound on the set of routes considered to be the quasi-optimal routing system does not include the effects of the bus capacity and riding time constraints, it does include a partial effect of the number of routes included in the system. However, it provides some measure for assessing the quality of the set of routes developed even though it may be unattainable.

Thus, this heuristic procedure is considered to have a sound basis.

B. Computational Experience

Since there is no known bus routing method suitable for use on a computer which will guarantee an optimum solution, any heuristic procedure must be judged not only with respect to its degree of success relative to the best set of routes available for known problems but also with respect to its consistency in the level of success.

A computer program based upon the procedure described in this dissertation was written in FORTRAN IV for use on the CDC 6400 computer. It can handle bus routing for a school district involving any number of periods, nine route origins and nine schools per period, one hundred twenty stops per school, thirty-four routes per school, and thirty stops per route. All computations are done in integer arithmetic.

Computational experience was gained in three phases. First, Newton's procedure was applied to school bus routing and delivery problems reported in the literature. Then, the relationship between the amount of computer time required to develop a routing system and the number of stops serviced by the set of routes was examined. Finally, this method was used to develop a set of bus routes for four schools in the Williamsville Central School District, a suburban area in Western New York.

Thompson[34] drew a map of a hypothetical school system showing the location of the school, the location of 31 bus stops specified by the school, and the number of students assigned to each stop. Moreover, all roads were marked off in units of one-half mile. In order to simulate the area of a real school system as closely as possible, the map included features such as: isolated areas, contour roads, and varying population densities. The elements of the symmetric interstop distance matrix calculated from this map ranged between 1 and 13.5 miles; the elements of the student load vector varied between 3 and 22 students. Although it was stated that 3 minutes were required to travel one mile and that loading at each bus stop required a minute, no restriction was imposed on the student riding time. Buses of 30, 36, 42, 48, 54, 60, 66, and 72 passenger capacities were available. The problem was to design the set of bus routes, all of which started and ended at the school, required to transport the 252 students assigned to the 31 bus stops.

The group of 49 school superintendents with experience in school bus routing and the group of 37 transportation directors who participated in Thompson's study were allowed to design the routing system for the hypothetical school by any manual method they favored.

The best set of routes developed by a member of the group of transportation directors involved 5 routes whose loads varied between 28 and 67 students and whose total traveling distance was 100 miles. The best set of routes designed by a member of the group of school superintendents consisted of 5 routes whose loads ranged between 33 and 67 students and whose total traveling distance was 101 miles.

Newton's method was applied to the same hypothetical school for all combinations of 4 bus capacities and 4 maximum allowable riding distances or a total of 16 cases. Each case was run until Algorithm A was exhausted at iteration or pass 49, i.e. the Nearest City Approach generated 31 trial routes and Algorithm A generated 17 "infinite capacity" bus routes. 177.133 seconds of computer time were used to develop the routing systems for the 16 cases or approximately 11 seconds per case. Most of the solutions were accepted from one of the last 3 trial routes generated by Algorithm A as was expected. A summary of the results appears in Table 1.

Case 13 defined by a bus capacity of 54 students and a maximum riding distance of 20 miles yielded the best routing system whose total traveling distance was 94.5 miles. This represents a savings of 5.8% miles with respect to the best set of routes developed by a transportation director participating in Thompson's study[34]. The best set of routes for Case 13 is:

THOMPSON DATA

31 STOPS SERVICED
252 STUDENTS TRANSPORTED
APPROXIMATELY 11 SECONDS CDC 6400 COMPUTER TIME PER CASE

CASE	CAPACITY OF BUS	MAXIMUM RIDING DISTANCE	MINIMUM NUMBER ROUTES	NUMBER ROUTES USED	DISTANCE ROUTE SYSTEM	RATIO	NUMBER ANSWER PASS
1	72	20.0 mi.	4	5	97.0 mi.	1.48	44
2	72	18.0	4	5	97.0	1.48	47
3	72	17.5	4	5	95.5	1.46	47
4	72	17.0	4	5	95.5	1.46	48
5	66	20.0	4	5	97.0	1.48	44
6	66	18.0	4	5	97.0	1.48	47
7	66	17.5	4	5	95.5	1.46	47
8	66	17.0	4	5	95.5	1.46	48
9	60	20.0	5	5	95.5	1.46	44
10	60	18.0	5	5	95.5	1.46	47
11	60	17.5	5	6	103.5	1.54	47
12	60	17.0	5	6	106.0	1.58	46
13	54	20.0	5	5	94.5	1.44	44
14	54	18.0	5	6	102.0	1.52	47
15	54	17.5	5	6	102.0	1.52	47
16	54	17.0	5	6	105.5	1.57	47

Table 1

Number	Route	Load	Miles
1	School-O-U-P-R-Q-RR-S-School	50	17.5
2	School-V-T-Z-W-School	50	16.0
3	School-Y-F-E-A-B-D-C-DD-School	50	23.0
4	School-X-G-H-HH-I-J-School	53	21.5
5	School-OO-K-LL-M-L-N-School	49	16.5

Cases 3, 4, 7, 8, 9 and 10 produced routing systems whose total traveling distance was 95.5 miles. However, all the routing systems were not identical. Cases 3, 4, and 7 resulted in the same routing system; case 8 yielded a second set of routes; cases 9 and 10 produced a third routing system. Thus, by developing routing systems for various combinations of bus capacity and passenger riding distance constraints and a fixed interstop distance matrix it may be possible to produce alternative routing systems whose total traveling distance is identical. A routing system can then be selected from these sets upon the basis of either bus load or individual route length variability or some other statistic considered important by the school transportation personnel. Studying alternate optimal solutions can be easily accomplished by using this computational procedure because of its speed and efficiency.

Boyer[6] designed a set of bus routes, all of which start and end at the school, to transport 575 students assigned to 45 bus stops for a school in Hennepin County, Minnesota. The problem involved a symmetric interstop travel time matrix, a bus capacity constraint of 65 students, and a riding time constraint of infinity. Since Boyer's method does not require knowledge of all the elements of the interstop travel time matrix, only 171 elements were listed in (6). The elements of the student load vector ranged between 1 and 33 students; the known elements of the interstop travel time matrix varied between 1 and 23 minutes. Although the absolute minimum number of buses needed to provide transportation for the school was 9, Boyer developed a routing system using 11 buses and requiring 413 minutes traveling time.

Newton's procedure was applied to the same problems until Algorithm A was exhausted. At iteration 51, i.e. the Nearest City Approach generated 45 trial routes and Algorithm A generated 5 trial routes. A routing system using 10 buses whose total traveling time was 394 minutes with a lower bound of 221 minutes was developed in 13.392 seconds. The quasi-optimal routing system, obtained from iteration 49 of this procedure, requires 4.8% fewer minutes than Boyer's solution. The author feels that a better solution would have been obtained with the complete interstop travel time matrix. The best set of routes is:

Number	Route	Load	Time
1	S-22-23-S	60	32
2	S-24-25-26-31-21-20-19-S	65	38
3	S-37-41-36-35-S	65	27
4	S-40-39-34-38-S	60	30
5	S-33-32-29-30-27-28-12-S	57	62
6	S-6-13-11-5-1-S	63	49
7	S-2-3-4-7-9-8-S	56	48
8	S-10-43-42-S	62	36
9	S-44-45-18-17-S	61	36
10	S-16-15-14-S	26	36

Thus, Newton's method was able to obtain better routing systems for the only two school bus routing problems which have appeared in the literature. The results for Thompson's data[34] were especially encouraging because experienced transportation directors can usually develop almost optimum routing systems for problems involving 20-30 stops by visual trial-and-error adjustment.

The rest of the problems from the literature which were solved by this computational procedure are delivery problems. Although, the school bus scheduling problem and the delivery problem are conceptually the same, differences which exist in them must be considered when evaluating a heuristic procedure.

The school bus scheduling problem is characterized by a large number of stops which must be serviced and a non-symmetric interstop travel time matrix for reasons such as: restrictions imposed upon the crossing of busy streets by students, one-way streets and limited access highways. The elements of the interstop time matrix usually have a narrow range and a low variability because of the relatively small area serviced by a school and the restrictions imposed upon student walking time to the bus stop. The elements of the student load vector also have a narrow range and a low variability because of upper bounds usually placed upon the number of students assigned to a bus stop to reduce the noise level and the possibility of landscape damage in residential areas. Moreover, the bus stops are usually arranged in groups because of housing developments and the placement of bus stops along main thoroughfares in sparsely populated areas. Even in sparsely populated areas, no bus stop is really isolated, i.e. each stop is a relatively short distance from either the school or any other stop designated by the school. Moreover, any computer based procedure for the school bus scheduling problem must require little computer time in order to be of practical value to a school district because of the frequent updating of routing systems necessitated by population fluctuations, changes in school boundaries, and the building of new schools and lack of funds.

The delivery problem is characterized by relatively few stops and a symmetric interstop distance matrix whose elements usually have a wide range and high variability because of the large area serviced by a warehouse or depot. Since no restrictions are placed upon customer demands, the elements of the customer demand vector also have a wide range and high variability. Moreover, the delivery stops are not necessarily arranged in groups because customer demand is not area dependent. Then too, any computer based procedure for the delivery problem may require a great deal of computer time and still be acceptable to a corporation which normally allocates ample funds for the development of routing systems.

These differences between the delivery problem and the school bus scheduling problem are great enough to make the widespread interchange of heuristic solution procedures infeasible. Therefore, an efficient computational procedure which was designed primarily to handle routing problems possessing the characteristics of the school bus scheduling problem is not expected to be consistently superior with respect to the quality of the routing systems developed when applied to delivery problems.

Dantzig and Ramser[14] developed a set of truck routes, all of which start and end at the depot, to deliver 18200 gallons of material to 12 customers in 6000 gallon capacity trucks. The elements of the symmetric interstop distance matrix varied between 5 and 52 units; the elements of the customer demand vector ranged between 1100 and 1900 gallons.

The Dantzig and Ramser method produced a routing system using 4 routes whose total traveling distance was 294 units.

Clarke and Wright[9] developed a routing system for the same problem requiring 4 routes covering 290 units, the conjectured optimum.

Newton's procedure was applied to the same delivery problem until Algorithm A was exhausted at interation 13. A routing system using 4 routes whose total traveling distance was 304 distance units was developed in 1.117 seconds. The solution, obtained from iteration 5 of this procedure, requires 3.4% more distance units than the Dantzig and Ramser solution and 4.8% more distance units than the conjectured optimum.

The set of routes developed by this procedure is:

Number	Route	Load	Distance
1	0-6-7-5-0	4300	64
2	0-9-8-10-0	5300	92
3	0-11-12-4-3-0	5700	120
4	0-2-1-0	2900	28

Clarke and Wright[9] designed a set of truck routes, all of which start and end at the depot, to deliver 104,300 pounds of goods to 30 customers in 14,000 pound capacity trucks. The elements of the symmetric interstop distance matrix ranged between 3 and 98 miles; the elements of the customer demand vector varied between 100 and 12,300 pounds. The Clarke and Wright solution[9] used 8 trucks, the absolute minimum number of trucks possible, and required 1427 miles. For the same problem, the Dantzig and Ramser method[14] produced a routing system using 10 routes whose total traveling distance was 1766 miles. By visual trial-and-error adjustment, Gaskell[18], developed a routing system for this problem using 8 routes and covering 1416 miles.

Newton's procedure was applied to the same problem until Algorithm A was exhausted at iteration 44. A routing system using 9 routes whose total traveling time was 1544 miles was developed in 7.948 seconds. The solution, obtained from iteration 5, required 9% more miles than the Gaskell solution, 8.2% more miles than the Clarke and Wright method, and 14.3% fewer miles than the Dantzig and Ramser solution.

The set of routes developed by this method is:

Number	Route	Load	Distance
1	0-6-5-11-16-15-9-7-13-29-0	13700	215
2	0-12-14-4-3-24-22-23-0	13500	207
3	0-27-26-0	11900	199
4	0-8-10-19-0	8700	176
5	0-18-25-20-0	12200	150
6	0-2-1-21-17-0	8500	111
7	0-30-0	12300	136
8	0-28-0	9500	186
9	0-19-0	14000	164

Cochran[10] designed the routing systems for two delivery problems using a modified Clarke and Wright method. Problem 1 involved 14,461 units to be delivered to 12 customers in trucks of 4500 unit capacity. The elements of the symmetric interstop distance matrix varied between 8 and 315 units; the elements of the customer demand vector ranged between 100 and 3726 units. The Cochran solution[10] used 4 routes whose total traveling distance was 1433 units.

Newton's method was applied to Problem 1 until Algorithm A was exhausted at iteration 19. A routing system containing 4 routes whose total traveling distance was 1383 units was developed in 1.717 seconds. The solution, obtained from iteration 18 of this procedure, required 3.6% fewer distance units than the solution produced by the modified Clarke and Wright method. Although the relative locations of the delivery stops are unavailable, the author feels that this computational procedure produced a superior solution because the delivery stops were arranged in groups and thus Problem 1 resembled a school bus scheduling problem.

The set of routes developed by this method is:

Number	Route	Load	Distance
1	0-7-8-5-0	3290	185
2	0-14-0	3726	444
3	0-6-12-13-11-10-0	3745	478
4	0-4-3-2-9-0	3700	276

The first two routes appeared in both routing systems.

Cochran's Problem 2[10] involved 1405 units of goods to be delivered to 25 customers in 120 unit capacity trucks. The elements of the symmetric interstop distance matrix varied between 2 and 221 distance units; the elements of the customer demand vector ranged between 15 and 100 units. Using a modified Clarke and Wright method, Cochran designed a routing system using 14 trucks whose total traveling distance was 1468 distance units.

This computational procedure was applied to Problem 2 until Algorithm A was exhausted at iteration 31. A routing system using 14 trucks whose total traveling distance was 1486 distance units was developed in 3.637 seconds. The solution, obtained from iteration 12 of this procedure, travels 1.2% more distance units than the solution obtained by the modified Clarke and Wright method. Both solutions used 2 more trucks than the absolute minimum number of trucks possible. This was due to the combination of customer demand loads, most of which were greater than 50 demand units, and the location of the customers, i.e. customers in the same general area had total demands which exceeded the truck capacity and thus one route could service only one or two stops.

The set of routes developed by this method is:

Number	Route	Load	Distance
1	0-13-11-0	90	82
2	0-7-15-0	120	70
3	0-16-14-0	120	102
4	0-2-0	60	4
5	0-3-0	80	10
6	0-4-9-0	110	56
7	0-5-0	90	28
8	0-8-6-0	115	76
9	0-10-0	60	48
10	0-12-0	90	54
11	0-17-18-0	120	185
12	0-19-20-21-0	110	210
13	0-24-25-0	120	276
14	0-23-22-26-0	120	285

Routes 1, 5, 6, 7, 10, 11, 12, 13, and 14 appeared in both routing systems. If routes 4 and 9 are merged into one route 0-2-10-0 which requires 52 distance units, then the routing system will use one less truck. However, the total traveling distance of the routing system with the merged routes in this case happens to remain unchanged. Although the purpose of a computer based routing procedure is to avoid visual trial-and-error adjustments which become less efficient and highly impractical as problem size increases, this merging of two routes is noted as a point of interest.

Gaskell[18] created four new delivery problems which were constrained by both truck capacity and route length. Total route distance included not only the number of units covered in traveling between delivery stops but also an allowance of 10 miles for each customer serviced. These problems were designed to compare the efficiency of variations of the Clarke and Wright method with respect to groupings of stops and isolated customers.

Routing systems for each of these delivery problems were developed by the Visual Method, a combination of trial-and-error adjustment and manual permutation of groups of delivery stops. The routing systems produced by the Visual Method were the best available and therefore were used to evaluate the routing systems developed by the computer based variations of the Clarke and Wright method. Since these problems involved at most 32 delivery stops, the time consuming Visual Method produced optimal routing systems. However, for larger problems it would be unable to compete with computer based procedures. Thus, the routing systems developed by the computational procedure of this dissertation will be compared primarily with the routing systems produced by variations of the Clarke and Wright method, a computer based procedure.

The first problem, Case Study Number 3, involved 29,370 units to be delivered to 32 customers in 8000 unit trucks. Each route was not to exceed 240 miles including the mileage allowance of 10 miles per customer. The elements of the interstop distance matrix varied between

1 and 118 miles; the elements of the customer demand vector ranged between 40 and 4000 units. This case is characterized by a close grouping of some of the customers, a centrally located depot with respect to 30 of the delivery stops and 2 customers located a great distance from the depot.

The Visual Method yielded a routing system using 4 routes whose total traveling distance was 813 miles. The best solution produced by a variation of the Clarke and Wright method used 5 routes and covered 821 miles. However, the poorest solution produced by a variation of the Clarke and Wright method used 5 routes and required 850 miles.

Newton's procedure was applied to Case Study Number 3 until Algorithm A was exhausted at iteration 39. It produced a routing system using 5 routes whose total traveling distance was 886 miles. The solution, obtained from iteration 23 of the procedure, required 7.9% more miles than the best solution obtained by a variation of the Clarke and Wright method. The reason that this computational procedure designed a less desirable routing system was the presence of the isolated delivery stops which are generally not present in the school bus scheduling problem.

The set of routes developed by this method is:

Number	Route	Load	Distance
1	0-17-24-23-22-20-21-18-19-15-14-0	6900	228
2	0-1-11-5-6-7-8-9-10-32-13-0	7920	177
3	0-31-30-3-4-2-12-0	6350	222
4	0-29-28-27-26-25-0	7500	147
5	0-16-0	700	112

The second problem, Case Study Number 4, involved 22,500 units to be delivered to 21 customers in 6000 unit trucks. Each route was to be less than 200 miles in length including the mileage allowance of 10 miles per customer. The elements of the interstop distance matrix varied between 3 and 83 miles; the elements of the customer demand vector ranged between 100 and 2500 units. None of the customers are isolated from either the depot or from other customers. The variability of the elements of the interstop distance matrix is relatively low.

The Visual Method produced a routing system using 4 routes whose total traveling distance was 585 miles. The best solution produced by a variation of the Clarke and Wright method used 4 routes and covered 598 miles. However, the poorest solution produced by a variation of the Clarke and Wright method used 4 routes and required 648 miles.

Newton's procedure was applied to Case Study Number 4 until Algorithm A was exhausted at iteration 25. A routing system using 4 routes whose total traveling distance was 593 miles was developed. The solution, obtained from iteration 22 of this procedure, required 1% fewer miles than the best and 9% fewer miles than the poorest routing systems designed by variations of the Clarke and Wright method.

The set of routes developed by this method is:

Number	Route	Load	Distance
1	0-6-1-2-5-7-9-0	5600	173
2	0-10-8-3-4-11-13-0	5400	162
3	0-12-15-18-16-14-0	5500	126
4	0-17-20-21-19-0	6000	132

Routes 1 and 2 also appear in the routing system developed by the Visual

Method. Since none of the customers were isolated and the variability of the elements of the interstop distance was low, Case Study Number 4 possessed two of the characteristics of the school bus scheduling problem and this computational procedure developed a superior routing system.

The third problem, Case Study Number 5, involved 12,750 units of goods to be delivered to 29 customers in 4500 unit capacity trucks. Each route was not to exceed 240 miles including the mileage allowance of 10 miles per delivery stop. The elements of the interstop distance matrix ranged between 1 and 121 miles; the elements of the customer demand vector varied between 100 and 3100 units. This problem was characterized by a loose grouping of customers located at various distances from the depot. Gaskell considered this to be a difficult problem.

The Visual Method designed a routing system containing 4 routes whose total traveling distance was 876 miles. The best solution produced by a variation of the Clarke and Wright method used 5 routes and required 943 miles. However, the poorest routing system developed by a variation of the Clarke and Wright method involved 5 routes covering 1017 miles.

Newton's method was applied to Case Study Number 5 until Algorithm A was exhausted at iteration 33. A routing system using 5 routes whose total traveling distance was 913 miles was developed. The solution, obtained from iteration 30 of this procedure, required 3.3% fewer miles than the best and 11.3% fewer miles than the poorest routing systems designed by variations of the Clarke and Wright method.

The set of routes developed by this method is:

Number	Route	Load	Distance
1	0-21-14-8-9-17-12-11-10-28-18-0	4125	220
2	0-15-16-7-13-0	1000	156
3	0-26-28-27-25-24-29-0	2850	234
4	0-3-6-1-4-5-2-0	3975	216
5	0-22-20-19-0	800	87

Again, this computational method produced a superior routing system because the customers were arranged loosely in groups, a characteristic of the school bus scheduling problem.

The fourth problem, Case Study Number 6, involved 10,189 units of goods to be delivered to 22 customers in 4500 unit capacity trucks. Each route was not to exceed a length of 240 miles including the mileage allowance of 10 miles per delivery. The elements of the interstop distance matrix ranged between 4 and 145 miles; the elements of the customer demand vector varied between 60 and 4100 units. Some of the customers are loosely arranged in groups with the distance between customers greater than the distance between neighboring stops in other problems. One customer is isolated at a relatively great distance from the depot.

The Visual Method produced a routing system involving 5 routes whose total traveling distance was 949 miles. The best solution produced by a variation of the Clarke and Wright method used 5 routes and required 955 miles. However, the poorest solution produced by a variation of the Clarke and Wright method required 6 routes covering 1015 miles.

Newton's procedure was applied to Case Study Number 6 until Algorithm A was exhausted at iteration 25. A routing system involving 6 routes whose total traveling time was 1009 miles was developed. The solution, obtained from iteration 14 of this procedure, requires 5.7% more miles than the best solution produced by a variation of the Clarke and Wright method.

The set of routes developed by this method is:

Number	Route	Load	Distance
1	0-14-17-15-16-3-2-0	1144	227
2	0-11-13-6-1-0	775	156
3	0-10-0	4100	78
4	0-12-9-5-4-8-7-0	2700	200
5	0-18-19-22-20-0	1295	216
6	0-21-0	175	132

Routes 2, 3, and 6 appeared in both this routing system and the one developed by the Visual Method. 17.406 seconds of computer time were used to solve the last four problems. Again, as a point of interest, if routes 2 and 6 are merged to form the route 0-21-11-13-6-1-0 which requires 275 miles, then the routing system will contain one less route and the total traveling distance will be reduced to 996 miles.

Newton's procedure, developed primarily to handle the school bus scheduling problem, behaved as was to be expected when applied to eight delivery problems from the literature. When the delivery problem possessed some of the characteristics of the school bus scheduling problem, this computational method produced a superior routing system. On the other hand, when the delivery problem involved isolated customers and many customers who were not even loosely grouped in areas, then this computational method produced a less desirable routing system than the other computer based methods. However, the number of extra miles required by the routing systems developed by this computational procedure never exceeded the best solution generated by a variation of the Clarke and Wright method by more than 8.2% which occurred in the Clarke and Wright 30 stop problem[9]. Thus, from a practical standpoint, this computational procedure performed at an acceptable level even when applied to delivery problems possessing characteristics which the method was not designed to handle.

Since the ratio of the total traveling time or distance required by the set of routes developed divided by the lower bound on the total traveling time or distance is used not only to assess the efficiency of the routing system but to form a criterion for the acceptance or rejection of a set of routes as the quasi-optimal solution for the problem being considered, an examination of this ratio with respect to the problems from the literature is warranted.

A summary of the results obtained by applying this computational procedure to two school bus scheduling and eight delivery problems from the literature is given in Table 2. The column labeled % gives the percentage by which the method of this dissertation either exceeded or improved the total number of time/distance units required by the best routing system developed by a computer based procedure for each problem being considered. In the case of the Thompson-31 stop problem[34], routing systems developed by manual methods were the only ones available.

By observation, the ratio of the total traveling time/distance for a routing system divided by the lower bound on the total traveling time/distance yields little information about the relative efficiency of a routing system developed by this computational procedure in

SUMMARY OF RESULTS FOR PROBLEMS IN THE LITERATURE

PROBLEM NAME	TOTAL TIME/DISTANCE ROUTE SYSTEM	ROUTE SYSTEM LOWER BOUND	RATIO	%
THOMPSON-31 STOPS	94.5	65.5	1.44	-5.8
BOYER-45 STOPS	394.0	221.0	1.78	-4.8
DANTZIG & RAMSER-12 STOPS	304.0	145.0	2.10	+4.8
CLARKE & WRIGHT-30 STOPS	1544.0	868.0	1.78	+8.2
COCHRAN-12 STOPS	1383.0	816.0	1.69	-3.6
COCHRAN-25 STOPS	1486.0	383.0	3.88	+1.2
GASKELL #3-32 STOPS	886.0	620.0	1.43	+7.9
GASKELL #4-21 STOPS	593.0	432.0	1.38	-1.0
GASKELL #5-29 STOPS	913.0	590.0	1.55	-3.3
GASKELL #6-22 STOPS	1009.0	626.0	1.61	+5.7

Table 2

comparison with one generated for the same problem by another computer based method or the conjectured optimum routing system attainable.

The ratio seems to be sensitive to the range and variability of the elements of the interstop travel time/distance matrix and the grouping of stops as evidenced by the Thompson-31 stop problem[34] and the Gaskell #4-21 stop problem[18]. Both of these problems had inter-stop travel time/distance matrices whose elements had a narrow range and low variability and the stops were arranged in groups. Moreover, both of these problems also had low ratios.

This ratio appears to reflect the effect of using more than the absolute minimum number of routes required as evidenced by the Cochran-25 stop problem[10]. The routing system developed for the problem contained two more than the absolute minimum number of routes required and the ratio was 3.88.

The routing system developed by Newton's procedure for the Dantzig and Ramser-12 stop problem[14] traveled 4.8% more distance units than the conjectured optimal routing system whose total traveling distance was 290 distance units and the ratio was 2.10. However, the ratio of the total traveling distance for the conjectured optimal routing system divided by the lower bound on the total distance equals 2.0.

Although the ratio is a relatively poor predictor of the degree of optimality attained by a routing system developed by any method, it can still be used as a criterion for terminating this computational procedure. Knowledge of reasonable ratios which can be expected for a routing system developed for a school will be acquired after experience with this procedure has been gained.

Although this computational procedure has proved to be efficient with respect to the quality of routing systems developed and with respect to the amount of computer time used to solve ten problems from the literature, no problem involved more than 45 stops, the Boyer data[6]. A thorough evaluation of a heuristic procedure requires that its efficiency be also examined for problems of the size which it would be expected to handle in the real world.

Thus, 18 problems involving up to 120 bus stops were created from tables of random numbers to examine the computer time required to develop routing systems for problems involving various numbers of stops. The elements of the interstop travel time matrices ranged between 2 and 20 minutes; the elements of the student load vector varied between 1 and 9. These ranges of values were selected because they satisfied the characteristics associated with the school bus scheduling problem.

All problems were constrained by a student riding time of 45 minutes and all problems were run until Algorithm A was exhausted, i.e. no further trial routes could be generated. A summary of the results appears in Table 3.

By observation, it appears that the lower bus capacity for a problem involving the same number of bus stops requires slightly less computer time. This is due to the fact that the individual routes contain fewer bus stops than the routes for buses of larger capacity and the amount of time required to improve each individual route is decreased.

Moreover, it seems that the quasi-optimal routing system is usually selected from an iteration whose trial route was generated by Algorithm A. This evidence also appeared when routing systems were developed for ten problems from the literature and thus bears out the

SUMMARY OF SAMPLE SCHOOL BUS SCHEDULING PROBLEMS

NUMBER OF STOPS	CAPACITY OF BUS	NUMBER OF PUPILS	MINIMUM NUMBER ROUTES	NUMBER ROUTES USED	TIME ROUTE SYSTEM	TIME LOWER BOUND	RATIO	NUMBER LAST PASS	NUMBER SOLUTION PASS	COMPUTER TIME USED (seconds)
40	60	208	4	4	143	98	1.46	47	40	24.455
40	50	208	5	5	156	104	1.50	47	3	21.132
50	60	268	5	5	159	114	1.39	57	56	36.140
50	50	268	6	6	184	121	1.52	57	50	31.607
60	60	324	6	6	196	132	1.48	69	61	50.787
60	50	324	7	7	212	138	1.54	69	62	43.488
70	60	391	7	7	233	157	1.48	78	72	71.811
70	50	391	8	9	265	169	1.57	78	43	64.760
80	60	457	8	8	254	178	1.43	89	87	138.999
80	50	457	10	10	292	190	1.54	89	84	128.463
90	60	521	9	9	323	211	1.53	108	106	225.716
90	50	521	11	11	347	223	1.56	108	106	213.029
100	60	587	10	11	330	210	1.57	118	110	218.701
100	50	587	12	13	369	216	1.71	118	117	199.408
110	60	657	11	12	396	278	1.42	126	118	300.670
110	50	657	14	14	426	290	1.47	126	123	289.035
120	60	709	12	13	414	290	1.43	136	133	433.106
120	50	709	15	15	470	300	1.57	136	131	416.499

Table 3

conjecture that better trial routes tend to partition into better sets of routes.

Since, this computational procedure was able to develop routing systems for problems involving 120 stops in approximately seven minutes, it is considered to be efficient with respect to computer time usage and therefore would be of practical value to a school district.

Finally, to examine the worth of Newton's procedure with respect to a real world situation, it was applied to four schools in the Williamsville Central School District, a rapidly growing suburban area in Western New York.

This district maintains a fleet of 78 buses which service ten elementary schools, two middle schools and three secondary schools located within the 42 square mile area of the district and twenty private and special schools outside the district. Furthermore, it is anticipated that ten new schools will be added to the school system within the next five years. Therefore, the Williamsville Central School District expects to be continually faced with a complex school bus routing problem and expressed a great interest in using a practical computer based method for designing its school bus routes.

The transportation director and the assistant to the superintendent of schools requested that Newton's procedure be applied to Academy Elementary School, Forest Elementary School, South Senior High School, and Dodge Elementary School. The first three schools service the established, densely populated part of Williamsville. Dodge Elementary School is located in the new, sparsely populated section of the area.

For each school, the school administrators designated the bus stops on a large map of the area and assigned students to the stops from census tracts. Each bus stop was assigned two labels, a road code number and a map number.

The interstop travel distance matrices were then developed manually by using a map reader. Two people developed the matrices for the four schools in approximately forty hours. The elements of the interstop distance matrix were then multiplied by a time factor, specified by the school administrators, to convert the distance matrix into an interstop travel time matrix. This time factor was large enough to include the time spent in servicing a stop. Since the school administrators felt that variability in travel time was negligible when considering all the routing systems collectively, no attempt was made to include the effects of this variability.

The elements of the interstop travel time matrix for the Academy School varied between 0.5 and 14 minutes or 0.25 and 7 distance units; the elements of the student load vector ranged between 2 and 44. The large elements of the student load vector were due to the high density of the population in the area serviced by the Academy School and the lumping of points by the school administrators. The results of applying this computational procedure to the Academy School data is summarized in Table 4. The routing system developed by this heuristic procedure requires 40% fewer distance units and 1 bus less than the current routing system used by the school.

The elements of the interstop travel time matrix for the Forest School ranged between 0.5 and 14 minutes or 0.25 and 7 distance units; the elements of the student load vector varied between 1 and 30 students. The Forest School is located in a high population density area. The results of applying Newton's method to the Forest School

ACADEMY SCHOOL

TYPE OF AREA	-	densely populated
NUMBER OF STOPS	-	45 (including origin and terminus)
NUMBER OF STUDENTS	-	532
NUMBER OF ROUTE ORIGINS	-	1 (South Senior High School)
MAXIMUM RIDING TIME	-	45 minutes

CAPACITY OF BUS	MINIMUM NUMBER ROUTES	NUMBER OF ROUTES USED NEWTON	NUMBER OF ROUTES USED SCHOOL	DISTANCE UNITS ROUTE SYSTEM NEWTON	DISTANCE UNITS ROUTE SYSTEM SCHOOL	DISTANCE LOWER BOUND	RATIO	NUMBER LAST PASS	NUMBER ANSWER PASS	COMPUTER TIME USED (seconds)
72	8	9	-	61.00	-	28.50	2.14	64	63	39.990
67	8	9	10	63.75	89.0	28.50	2.24	64	61	36.482
62	9	10	-	67.00	-	28.75	2.33	64	44	38.496
57	10	11	-	80.75	-	29.00	2.78	64	3	31.740
52	11	12	-	77.50	-	29.25	2.65	64	60	31.815

1 distance unit = 0.4444 mile

Algorithm A was exhausted

Table 4

data is summarized in Table 5. The routing system developed by this computational procedure requires 38% fewer distance units than the current routing system used by the school. However, both routing systems use 10 buses when the bus capacity is 67 students. Because of the lumping of bus stops, the routing systems required more than one extra bus over the absolute minimum number of buses required when the bus capacity was reduced to 57 and 52.

The elements of the interstop travel time matrix for the South Senior High School varied between 0.5 and 22 minutes or 0.25 and 11 distance units; the elements of the student load vector ranged between 1 and 41 students. These large student load elements were due to lumping of bus stops by the school administrators. The results of applying this computational procedure to the South Senior High School data is summarized in Table 6. The routing system developed by Newton's procedure requires 25% fewer distance units and one less bus than the current routing system used by the school.

The elements of the interstop travel time matrix for the Dodge School varied between 0.5 minute and 35 minutes or 0.25 and 17.5 distance units; the elements of the student load vector ranged between 1 and 27 students. All the routes for the Academy School started from the South Senior High School, all the routes for the Forest School started from the Academy School and all the routes for the South Senior High School started from the garage. However, the bus routes for the Dodge School can start at any of six origins. The results of applying Newton's procedure to the Dodge School data is summarized in Table 7. The routing system used by this heuristic procedure requires 17% fewer distance units and two less buses than the current routing

FOREST SCHOOL

TYPE OF AREA	-	densely populated
NUMBER OF STOPS	-	37 (including origin and terminus)
NUMBER OF STUDENTS	-	596
NUMBER OF ROUTE ORIGINS	-	1 (Academy School)
MAXIMUM RIDING TIME	-	45 minutes

CAPACITY OF BUS	MINUMUM NUMBER ROUTES	NUMBER OF ROUTES USED NEWTON	NUMBER OF ROUTES USED SCHOOL	DISTANCE UNITS ROUTE SYSTEM NEWTON	DISTANCE UNITS ROUTE SYSTEM SCHOOL	DISTANCE LOWER BOUND	RATIO	NUMBER LAST PASS	NUMBER ANSWER PASS	COMPUTER TIME USED (seconds)
72	9	9	-	57.00	-	26.75	2.13	45	26	8.807
67	9	10	10	63.25	87.0	27.75	2.28	45	32	6.863
62	10	11	-	66.00	-	28.75	2.30	45	4	7.116
57	11	13	-	72.75	-	30.75	2.37	45	12	7.407
52	12	15	-	84.00	-	32.75	2.56	45	31	7.019

1 distance unit = 0.4444 mile

Algorithm A was exhausted

Table 5

SOUTH SENIOR HIGH SCHOOL

TYPE OF AREA	-	densely populated
NUMBER OF STOPS	-	76 (including origin and terminus)
NUMBER OF STUDENTS	-	1097
NUMBER OF ROUTE ORIGINS	-	1 (garage - Mill Middle School)
MAXIMUM RIDING TIME	-	45 minutes

CAPACITY OF BUS	MINIMUM NUMBER ROUTES	NUMBER OF ROUTES USED		DISTANCE UNITS ROUTE SYSTEM		DISTANCE LOWER BOUND	RATIO	NUMBER LAST PASS	NUMBER ANSWER PASS	COMPUTER TIME USED (seconds)
		NEWTON	SCHOOL	NEWTON	SCHOOL					
65	17	19	20	175.0	219.5	72.25	2.42	97	92	136.883
60	19	21	-	189.0	-	75.25	2.51	97	89	137.229
55	20	24	-	211.5	-	79.75	2.65	97	87	137.248
50	22	28	-	233.5	-	85.75	2.72	97	45	135.208
45	25	30	-	251.0	-	88.75	2.83	97	81	134.462

1 distance unit = 0.4444 mile

Algorithm A was exhausted

Table 6

DODGE SCHOOL

TYPE OF AREA	-	sparsely populated
NUMBER OF STOPS	-	96 (including origin and terminus)
NUMBER OF STUDENTS	-	669
NUMBER OF ROUTE ORIGINS	-	6
MAXIMUM RIDING TIME	-	45 minutes

CAPACITY OF BUS	MINIMUM NUMBER ROUTES	NUMBER OF ROUTES USED		DISTANCE UNITS ROUTE SYSTEM		DISTANCE LOWER BOUND	RATIO	NUMBER LAST PASS	NUMBER ANSWER PASS	COMPUTER TIME USED (seconds)
		NEWTON	SCHOOL	NEWTON	SCHOOL					
72	10	10	-	196.50	-	107.5	1.81	100	100	90.999
72	10	10	-	196.50	-	107.5	1.81	105	102	148.213
72	10	10	-	196.50	-	107.5	1.81	110	102	243.488
67	10	11	13	204.00	238.5	116.5	1.78	100	82	60.989
67	10	11	13	204.00	238.5	116.5	1.78	105	82	116.821
67	10	11	13	204.00	238.5	116.5	,78	110	82	211.202
62	11	12	-	221.50	-	121.5	1.82	100	31	70.240
62	11	12	-	221.00	-	121.5	1.32	105	102	126.550
62	11	12	-	221.00	-	121.5	1.82	110	102	218.009
57	12	13	-	239.25	-	131.0	1.83	100	86	63.323
57	12	13	-	238.00	-	131.0	1.82	105	101	119.392
57	12	13	-	234.25	-	131.0	1.79	110	109	213.476

1 distance unit = 0.444 mile

Algorithm A was not exhausted

Table 7

system used by the school. Since this problem involved 96 stops, a study was made to examine the effect of limitations upon the number of trial routes generated. By observation, it can be noted in Table 7 that increasing the number of iterations allowed for each case used considerably more computer time but did not usually reduce the total number of distance units required by the routing system.

The application of Newton's computational procedure to four actual school bussing problems demonstrated its effectiveness. It was able to develop routing systems which were superior to the routing systems currently used by these schools with respect to the distance traveled and the number of buses used. Moreover, it was able to produce these high quality routing systems using an operationally acceptable amount of computer time.

VII. SUMMARY AND RECOMMENDATIONS

The model developed in this dissertation is a general one which utilizes variables that are applicable to all school systems. It represents a significant improvement over the manual and other computer based methods available in that it generates routing systems which are efficient with respect to the total mileage traveled and the number of routes required using a minimal amount of computer time. Moreover, it routes buses from school-to-school in addition to developing the individual routes for a school.

A review of the work accomplished during this study leads to the following recommendations for future efforts in the automatic design of school bus routes:

1. The manual calculation of the elements of the interstop distance matrix is tedious, error prone, and time consuming. Therefore, a great need exists for a computer based procedure for developing an accurate, non-symmetric interstop distance matrix involving 50-120 bus stops located in either densely or sparsely populated areas.
2. A computer based procedure for determining a student load vector whose elements could be constrained by student walking distance and size of load at a stop would be useful.
3. In order to avoid the development of a routing system involving a combination of full bus loads and half filled buses and a combination of maximum length and very short routes, a need exists for a method which would balance the

sizes of the bus loads and the lengths of the individual routes.

4. Since government aid is usually given to school districts for every bus load of a certain size, it would be useful to be able to specify both upper and lower limits on bus capacity.

5. School districts are interested in the cost associated with a transportation system. Therefore, it would be useful to be able to develop routing systems with respect to minimizing a cost function which would include the effects of the total mileage traveled and the size of the buses used.

6. A great need exists for a computer based procedure which would optimally assign all the routes which a particular bus would service between the time it left the garage and returned.

VIII. BIBLIOGRAPHY

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

2. Barachet, L. L., "Graphic Solutions of the Traveling-Salesman Problem", Operations Research, Vol. 5, (1957), pp. 841-845.

3. Bellman, R., "Dynamic Programming Treatment of the Traveling-Salesman Problem", Journal of the Association of Computing Machines, Vol. 9, (1962), pp. 61-63.

4. Bellmore, M. and G. L. Nemhauser, "The Traveling-Salesman Problem: A Survey", Operations Research, Vol. 16, (1968), pp. 538-549.

5. Boyer, R. A., "The Use of Mathematical Programming to Solve Certain Problems in Public School Transportation," Cooperative Research Project 783, University of Mississippi, 1961.

6. Boyer, R. A., "The Use of a Computer to Design School Bus Routes", Cooperative Research Project 1605, University of Mississippi, 1964.

7. Boyer, R. A., R. D. Ross, and T. A. Ross, "The Use of a Computer to Design School Bus Routes: A Status Report", Journal of Educational Data Processing, Vol. 4, (1967), pp. 143-151.

8. Braun, W. C. E., "A Computerized Simulation Approach to the Solution of the Carrier Dispatching Problem", M.S. Thesis, Kansas State University, 1967.

9. Clarke, J. and J. W. Wright, "Scheduling of Vehicles From a Central Depot to a number of Delivery Points", Operations Research, Vol. 12, (1964), pp. 568-581.

10. Cochran, H. M., "Optimization of a Carrier Routing Problem", M.S. Thesis, Kansas State University, 1967.

11. Crocs, G. A., "A Method for Solving Traveling-Salesman Problems", Operations Research, Vol. 6, (1958), pp. 791-812.

12. Dantzig, G. B., D. R. Fulkerson, and S. M. Johnson, "Solution of a Large-Scale Traveling-Salesman Problem", Operations Research, Vol. 2, (1954), pp. 393-410.

13. Dantzig, G. B., D. R. Fulkerson, and S. M. Johnson, "On a Linear Programming Combinatorial Approach to the Traveling-Salesman Problem", Operations Research, Vol. 7, (1959), pp. 58-66.

14. Dantzig, G. B. and J. H. Ramser, "The Truck Dispatching Problem", Management Science, Vol. 6, (1959), pp. 80-91.

15. Davis, R. D., "On the Delivery Problem and Some Related Topics", Doctoral Dissertation, Northwestern University, 1968.

16. Fletcher, A., "Alternative Routes Round the Delivery Problem", Data and Control, Vol. 1, (1963), pp. 20-22.

17. Flood, M. M., "The Traveling-Salesman Problem", Operations Research, Vol. 4, (1956), pp. 61-75.

18. Gaskell, T. J., "Bases for Vehicle Fleet Scheduling", Operational Research Quarterly, Vol. 18, (1967), pp. 281-295.

19. Gaunt, S., "A Non-Computer Method Using Search for Resolving the Traveling-Salesman Problem", Canadian Operational Research Society Journal, Vol. 6, (1968), pp. 44-54.

20. Hadley, G., Linear Programming, Addison-Wesley Publishing Co., Inc., 1962.

21. Hayes, R. L., "The Delivery Problem", Doctoral Dissertation, Carnegie-Mellon University, 1967.

22. Held, M. and R. M. Karp, "A Dynamic Programming Approach to Sequencing Problems, SIAM Journal, Vol. 10, (1962), pp. 196-210.

23. Karg, R. L. and G. L. Thompson, "A Heuristic Approach to Solving Traveling-Salesman Problems", Management Science, Vol. 10, (1964), pp. 225-248.

24. Kuo, S. S. and W. K. Young, "Computer Studies of the Traveling-Salesman Problem", Quarterly Bulletin of the Canadian Information Society, Vol. 8, (1968), pp. 31-36.

25. Lawler, E. L. and D. E. Wood, "Branch-and-Bound Methods: A Survey", Operations Research, Vol. 14, (1966), pp. 699-717.

26. Lin, S., "Computer Solutions of the Traveling-Salesman Problem", The Bell System Technical Journal, Vol. 44, (1965), pp. 2245-2269.

27. Little, J. D. C., K. G. Murty, D. W. Sweeney, and C. Karel, "An Algorithm for Solving the Traveling-Salesman Problem", Operations Research, Vol. 11, (1963), pp. 972-989.

28. Miller, C. E., A. W. Tucker, and R. A. Zemlin, "Integer Programming Formulation of Traveling-Salesman Problems", ACM Journal, Vol. 7, (1960), pp. 326-329.

29. Newton, R. M., "A School Bus Scheduling Algorithm", M.S. Thesis, State University of New York at Buffalo, 1967.

30. Newton, R. M. and W. H. Thomas, "Design of School Bus Routes by Computer", Socio-Economic Planning Sciences, Vol. 3, (1969), pp. 75-85.

31. Nicholoson, T. A. J., "A Boundary Method for Planar Traveling-Salesman Problems", Operational Research Quarterly, Vol. 19, (1968), pp. 445-452.

32. Reiter, S. and G. Sherman, "Discrete Optimizing", SIAM Journal, Vol. 13, (1965), pp. 864-889.

33. Roberts, S. M. and B. Flores, "An Engineering Approach to the Traveling-Salesman Problem", Management Science, Vol. 13, (1966), pp. 269-288.

34. Thompson, W. A., Jr., "The Use of Boyer's "Sequential Steps" for Routing School Buses", Doctoral Dissertation, University of Mississippi, 1963.

35. Tillman, F. A., "Dynamic Programming Solution to the School Bus Scheduling Problem", Unpublished Working Papers, Kansas State University, 1965.

36. Tillman, F. A. and H. Cochran, "A Heuristic Approach for Solving the Delivery Problem", The Journal of Industrial Engineering, Vol. 19, (1968), pp. 354-358.

APPENDIX

The Appendix contains a listing of Program BUS2 which generates bus routes and schedules for a multi-school system by means of the method described in this dissertation. Program BUS2 is written in FORTRAN IV for the CDC 6400 computer.

```
C     PROGRAM BUS2  -  RITA NEWTON                                          001
C     SELECT THE ORIGIN AND THE STOPS TO BE VISITED FOR EACH BUS ROUTE      002
      DIMENSION  IALPH(30), IARRT(30), ILOAD(30), INUM(30), IR1(120)        003
      DIMENSION  IREQ(35), IRTE(120), IRTE2(30), IRTE3(30), IRTP(240)       004
      DIMENSION  IVAIL(10), KDE(120), KL(35), KONRT(35), KORA(35)           005
      DIMENSION  KORN(35), KOST(10,35), KRT(35), KT(35), LAB(120)           006
      DIMENSION  LBA(10,35), LB01(10), LB02(10)                             007
      DIMENSION  LIST(30), LOAD(120), LOR(10), LSCH(6)                      008
C**** VECTOR  LSCH  CONTAINS  6  TEN-CHARACTER WORDS                 ****0802
C**** ALTER DIMENSION OF VECTOR  LSCH  AND ALL  READ/WRITE           ****0803
C**** STATEMENTS INVOLVING VECTOR  LSCH  ACCORDING TO THE            ****0804
C**** WORD LENGTH OF THE COMPUTER BEING USED                         ****0805
C**    **    **    **    **    **    **    **    **    **    **      **  0806
      DIMENSION  M(121,121), M2(31,31), MAXBUS(35), MINRT(155)              009
      DIMENSION  MOR(10,120), NBAV(10), NBUS(10,35), NBUS1(10,35)           010
      DIMENSION  NBUS2(10), NSTOP1(35), NSUPER(10)                          011
C     ARRAYS MUST BE DIMENSIONED AS FOLLOWS -                               012
C     LET  A = NUMBER OF STOPS ON INDIVIDUAL ROUTE (INCLUDES OR. + TER.)    013
C          B = NUMBER OF ROUTES FOR SCHOOL + 1                              014
C          C = NUMBER OF ORIGINS FOR PERIOD + 1                             015
C          D = NUMBER OF SCHOOLS SERVICED DURING PERIOD                     016
C          E = TOTAL NUMBER OF STOPS FOR SCHOOL ( INCLUDES OR. + TER.)      017
C     IALPH(A), IARRT(A), ILOAD(A), INUM(A), IR1(E), IREQ(B), IRTE(E)       018
C     IRTE2(A), IRTE3(A), IRTP(E+E), IVAIL(C), KDE(E), KL(B), KONRT(B)      019
C     KORA(B), KORN(B), KOST(C,B), KRT(B), KT(B), LAB(E), LBA(C,B)          020
C     LB01(C), LB02(C), LIST(A), LOAD(E), LOR(C), M(E+1, E+1)               021
C     M2(A+1, A+1), MAXBUS(B), MINRT(E+B ), MOR(C,E), NBAV(C), NBUS(C,B)    022
C     NBUS1(C,B), NBUS2(C), NSTOP1(B), NSUPER(D)                            023
C     PREPARE TAPE OR DISC  A  TO STORE SCHOOL DATA                         050
    1 REWIND 2                                                              051
C     READ LABEL FOR TIME PERIOD IDENTIFICATION                             053
      READ ( 5, 999 )                                                       054
  999 FORMAT ( 72H                                                          055
     1                      )                                               056
```

POOR ORIGINAL COPY - BEST AVAILABLE AT TIME FILMED

```
C     NOR = NUMBER OF POSSIBLE ORIGINS FOR BUS ROUTES DURING PERIOD
      READ ( 5, 1000 ) NOR
 1000 FORMAT ( 6X, 10I6 )
C     LOR(I) = 6 CHARACTER LABEL FOR ORIGIN I
C     NBAV(I) = NUMBER OF BUSES AVAILABLE AT ORIGIN I
      READ ( 5, 1001 ) ( LOR(I), NBAV(I), I = 1, NOR )
 1001 FORMAT ( 6X, A6 , I6 )
C     NBSUM = TOTAL NUMBER BUSES AVAILABLE
      NBSUM = 0
      DO 5 I = 1, NOR
      NBSUM = NBSUM + NBAV(I)
    5 CONTINUE
C     NSCH = NUMBER OF SCHOOLS SERVICED DURING PERIOD
      READ ( 5, 1000 ) NSCH
C     CLEAR MATRIX  NBUS  AND  LBA  FOR TRANSPORTATION ALGORITHM
      DO 9 I = 1, NOR
      DO 7 J = 1, NSCH
      NBUS(I,J) = 0
      LBA(I,J) = 0
    7 CONTINUE
    9 CONTINUE
C     K1 = COUNTER FOR NUMBER OF SCHOOLS PROCESSED
      K1 = 0
C     UPDATE K1
   17 K1 = K1 + 1
C     TEST - K1 = NSCH
      IF ( K1 - NSCH ) 21, 21, 125
C     LSCH(I) =  6 WORD VECTOR FOR SCHOOL IDENTIFICATION
C     N = NUMBER OF BUS STOPS, INCLUDES ORIGIN AND TERMINUS
C     KS = CODE FOR CONTROL OF INTERMEDIATE PRINT-OUT
C     KS = 0 MEANS NO INTERMEDIATE PRINT-OUT
C     KS = 1 MEANS EXECUTE INTERMEDIATE PRINT-OUT
C     KSS = CODE FOR THE SYMMETRY OF THE TIME MATRIX
C     KSS = 0 MEANS THE TIME MATRIX IS SYMMETRIC
```

POOR ORIGINAL COPY - BEST AVAILABLE AT TIME FILMED

```
C      KSS = 1 MEANS THE TIME MATRIX IS NOT SYMMETRIC
C      KPER = ALLOWABLE PERCENTAGE ABOVE LOWER BOUND FOR A SET OF ROUTES
C      KPASS = MAXIMUM NUMBER OF TRIAL ROUTES ALLOWED
    21 READ ( 5, 1002 ) ( LSCH(I), I = 1, 6  )
  1002 FORMAT ( 6X, 6A10 )
       READ ( 5, 1000 ) N, KS, KSS, KPER, KPASS
       NP1 = N+1
       NM1 = N-1
C      CLEAR MATRIX MOR
C      MOR(I,J) = NUMBER OF MINUTES REQUIRED TO TRAVEL FROM ORIGIN  I
C                 TO BUS STOP  J
       DO  24  I = 1, NOR
       DO  23  J = 1, N
       MOR(I,J) = 0
    23 CONTINUE
    24 CONTINUE
       DO  25  I = 1, NOR
       READ ( 5, 1000 ) ( MOR(I,J), J = 1, N )
    25 CONTINUE
C      CLEAR MATRIX M
C      M(I,J) = NUMBER OF MINUTES REQUIRED TO TRAVEL FROM
C               BUS STOP I  TO BUS STOP J
       DO  28  I = 1, N
       DO  27  J = 1, N
       M(I,J) = 0
    27 CONTINUE
    28 CONTINUE
C      TEST - MATRIX M IS SYMMETRIC
       IF ( KSS ) 39, 39, 49
C      READ LOWER HALF OF SYMMETRIC TIME MATRIX  M
C      READ ROWS 2 - N  BECAUSE ROW 1 WILL BE SELECTED FROM MATRIX  MOR
C      ROW N  IS REALLY COLUMN N  BECAUSE NO TRAVEL IS ALLOWED
C      BETWEEN THE TERMINUS OR SCHOOL AND ANY BUS STOP
    39 DO  40  I = 2, N
```

```
      READ ( 5, 1000 ) ( M(I,J), J = 1, I )                                112
   40 CONTINUE                                                             113
C     GENERATE THE UPPER HALF OF SYMMETRIC MATRIX  M                       114
      DO  44  I = 1, NM1                                                   115
      K = I+1                                                              116
      DO  43  J = K, N                                                     117
      M(I,J) = M(J,I)                                                      118
   43 CONTINUE                                                             119
   44 CONTINUE                                                             120
      GO TO 52                                                             121
C     READ NON-SYMMETRIC TIME MATRIX  M                                    122
C     READ ROWS 2 - N  BECAUSE ROW 1 IS SELECTED FROM MATRIX  MOR          123
   49 DO  50  I = 2, N                                                     126
      READ ( 5, 1000 ) ( M(I,J), J = 1, N )                                127
   50 CONTINUE                                                             128
C     LAB(I) = 6 CHARACTER LABEL FOR BUS STOP  I                           129
C     LOAD(I) = STUDENT LOAD ASSIGNED TO BUS STOP I                        130
   52 READ ( 5, 1001 ) ( LAB(I), LOAD(I), I = 1, N )                       131
C     MAXBL = MAXIMUM BUS LOAD                                             132
C     MAXRT = MAXIMUM ALLOWABLE RIDING TIME FOR STUDENT LOAD               133
C             ASSIGNED TO STOP 1  OF ANY ROUTE                             134
      READ ( 5, 1000 ) MAXBL, MAXRT                                        135
C     STORE INFINITY ELEMENTS IN COL.1, ROW N, DIAGONAL OF MATRIX  M       136
      DO  54  I = 1, N                                                     137
      M(I,I) = 9999                                                        138
      M(I,1) = 9999                                                        139
      M(N,I) = 9999                                                        140
   54 CONTINUE                                                             141
      M(1,N) = 9999                                                        1411
C     WRITE DATA FOR SCHOOL K1 ON TAPE OR DISC A                           142
      WRITE (2) K1, (LSCH(I), I=1,6 ), N, KS, KSS, KPER, KPASS,            143
     1 (  ( MOR(I,J), I=1,NOR ), J=1,N ), ( ( M(I,J), I=1,N ), J=1,N ),    144
     2 ( LAB(I), I=1,N ), ( LOAD(I), I=1,N ), MAXBL, MAXRT                 145
C     MINBUS = MINIMUM NUMBER OF BUSES REQUIRED BY SCHOOL K1               152
```

POOR ORIGINAL COPY - BEST
AVAILABLE AT TIME FILMED

```
C     MAXBUS(K1) = MAXIMUM NUMBER OF BUSES ALLOWED BY SCHOOL K1
   61 LSUM = 0
      DO  63  J = 2, NM1
      LSUM = LSUM + LOAD(J)
   63 CONTINUE
      ALSUM = LSUM
      AMAXBL = MAXBL
      AMIN = ( ALSUM / AMAXBL ) + 0.999999
      MINBUS = AMIN
C**** A USEFUL CHANGE WHEN ADJUSTING CONSTRAINTS                     ****
C**** IN ORDER TO OBTAIN A SOLUTION UNDER  TIGHT RESTRICTIONS         ****
C****               MAXBUS(K1) = MINBUS + KBUS                       ****
C***                WHERE  KBUS  IS AN INTEGER GREATER THAN ONE      ****
C**     **    **    **    **    **    **    **    **    **    **     **
      MAXBUS(K1) = MINBUS + 1
C     LBS = PORTION OF THE LOWER BOUND WHICH IS INDEPENDENT
C               OF THE ORIGIN AND THE NUMBER OF ROUTES USED
C     MATRIX  M(I,J)  IS ALTERED DURING THE PROCESS OF FINDING  LBS
C     KDE(J) = 0  MEANS COLUMN J  DOES NOT HAVE A ZERO
C     KDE(J) = 1  MEANS COLUMN J  HAS A ZERO
      DO 65 J = 1, N
      KDE(J) = 0
   65 CONTINUE
      LBS = 0
      DO  75  I = 2, NM1
      MIN = 77777
      DO  71  J = 2, N
      IF ( M(I,J) - MIN ) 69, 71, 71
C     MIN = SMALLEST ELEMENT IN ROW I
   69 MIN = M(I,J)
C     NCOL = NUMBER OF COLUMN IN WHICH SMALLEST ELEMENT LIES
      NCOL = J
   71 CONTINUE
C     UPDATE LBS AND KDE(NCOL)
```

```
      KDE(NCOL) = 1
      LBS = LBS + MIN
C     REDUCE ROW I  BY THE MINIMUM ELEMENT
      DO  73  J = 2, N
      M(I,J) = M(I,J) - MIN
   73 CONTINUE
   75 CONTINUE
C     LBO1(I) = LOWER BOUND WHEN ALL ROUTES START AT ORIGIN I
C                AND MINBUS ROUTES ARE USED
C     LBO2(I) = LOWER BOUND WHEN ALL ROUTES START AT ORIGIN I
C                AND MAXBUS(K1) ROUTES ARE USED
      DO  117  I = 1, NOR
C     MOVE ROW I  OF MATRIX MOR  TO ROW 1  OF MATRIX M
      DO  81  J = 1, N
      M(1,J) = MOR(I,J)
   81 CONTINUE
C     FIND PORTION OF LOWER BOUND CONTRIBUTED BY TRAVELING BETWEEN
C     ORIGIN I  AND THE  FIRST STOP OF ANY ROUTE
      MIN = 77777
      DO  89  J = 2, NM1
      IF ( M( 1,J ) - MIN ) 85, 89, 89
   85 NCOL = J
      MIN = M( 1, J)
   89 CONTINUE
      KDE( NCOL ) = 1
      LBO1(I) = LBS + ( MINBUS * MIN )
      LBO2(I) = LBS + ( MAXBUS(K1) * MIN )
C     REDUCE ROW 1  OF MATRIX M  BY  MIN
      DO  90  J = 2, NM1
      M( 1,J) = M(1,J) - MIN
   90 CONTINUE
      DO  113  J = 2, N
      IF ( KDE(J) ) 93, 93, 113
   93 MIN = 77777
```

```
      DO  101 K = 1, NM1
      IF ( M(K,J) - MIN ) 97, 101, 101
   97 MIN = M(K,J)
  101 CONTINUE
      IF ( J - N ) 105, 109, 109
  105 LB01(I) = LB01(I) + MIN
      LB02(I) = LB02(I) + MIN
      GO TO 113
  109 LB01(I) = LB01(I) + ( MINBUS * MIN )
      LB02(I) = LB02(I) + ( MAXBUS(K1) * MIN )
  113 CONTINUE
C     RESTORE VECTOR KDE TO STATUS OF STATEMENT 75
      KDE( NCOL ) = 0
  117 CONTINUE
C     WRITE LOWER BOUND DATA FOR SCHOOL K1  ON TAPE OR DISC A
      WRITE (2) ( LB01(I), I = 1, NOR ) , ( LB02(I), I = 1, NOR )
      WRITE (2) LBS, ( KDE(I), I=1,N ), ( ( M(I,J), I=1,N ), J=1,N )
C     TEST - NOR = 1
  120 IF ( NOR - 1 ) 17, 17, 121
C     LBA(I,K1) = AVERAGE LOWER BOUND OR EXPECTED MINIMUM COST OF ONE
C                 ROUTE STARTING AT ORIGIN I  FOR SCHOOL K1  WHEN
C                 MAXBUS(K1) BUSES ARE USED AND ALL ROUTES START
C                 AT ORIGIN I
  121 DO  123  I = 1, NOR
      LBA(I,K1) = ( LB02(I) * 100 ) / MAXBUS(K1)
  123 CONTINUE
      GO TO 17
C     AT 125 ALL  LOWER BOUNDS ETC. HAVE BEEN CALCULATED
C     PREPARE TAPE OR DISC  A  FOR READING SCHOOL DATA PREVIOUSLY STORED
  125 REWIND 2
C     COMPARE BUS AVAILABILITY AND BUS REQUIREMENTS
      ISUM = 0
      DO 126 I = 1, NSCH
      ISUM = ISUM + MAXBUS(I)
```

POOR ORIGINAL COPY - BEST AVAILABLE AT TIME FILMED

```
  126 CONTINUE
      IF ( ISUM - NRSUM ) 128, 128, 127
C     ERROR RETURN - BUS SHORTAGE
  127 WRITE ( 6, 1003 )
      WRITE ( 6, 999  )
      WRITE ( 6, 2000 ) ISUM, NRSUM
 2000 FORMAT ( 1H0, 5X,  5HERROR, 5X, I4, 14HBUSES REQUIRED,
     1 5X, I4, 15HBUSES AVAILABLE  )
      GO TO 1
C     TEST- NOR = 1
  128 IF ( NOR - 1 ) 139, 139, 129
C     EXECUTE TRANSPORTATION ALGORITHM
  129 CALL TRAN (NOR,NSCH,NRAV,MAXBUS,LRA,NBUS,KODE )
C     NBUS(I,J) = NUMBER OF BUSES WHICH START FROM ORIGIN I  AND
C                 TRAVEL TO TERMINUS OR SCHOOL J
C     NSUPER(J) = NUMBER OF THE SUPER-ORIGIN OR THE ORIGIN WHICH
C                SERVES THE GREATEST NUMBER OF ROUTES FOR SCHOOL J
      DO  137  J = 1, NSCH
      MAX = 0
      DO  135  I = 1, NOR
      IF ( NBUS(I,J) - MAX ) 135, 135, 133
  133 MAX = NBUS(I,J)
      NROW = I
  135 CONTINUE
      NSUPER(J) = NROW
  137 CONTINUE
      GO TO 143
C     AT 139 ALL SCHOOLS ARE SERVICED BY ORIGIN 1
  139 DO  141  J = 1, NSCH
      NSUPER(J) = 1
  141 CONTINUE
C     START PROCESSING EACH SCHOOL INDIVIDUALLY
C     K1 = COUNTER FOR NUMBER OF SCHOOLS PROCESSED
  143 K1 = 0
```

POOR ORIGINAL COPY - BEST AVAILABLE AT TIME FILMED

```
C       INTERMEDIATE PRINT-OUT OF PERIOD DATA
        IF ( KS ) 145, 145, 152
  152   WRITE ( 6, 1003 )
        WRITE ( 6, 999 )
        WRITE ( 6, 1029 )
 1029   FORMAT ( 1H0, 5X, 10HMATRIX LBA )
        DO 153 I = 1, NOR
        WRITE ( 6, 1020 ) I
        WRITE ( 6, 1021 ) ( LBA(I,J), J = 1, NSCH )
  153   CONTINUE
        WRITE ( 6, 1030 )
 1030   FORMAT ( 1H0, 5X,  4HNBUS )
        DO 154 I = 1, NOR
        WRITE ( 6, 1020 ) I
        WRITE ( 6, 1021 ) ( NBUS(I,J), J = 1, NSCH )
  154   CONTINUE
        WRITE ( 6, 1031 )
 1031   FORMAT ( 1H0, 5X,  6HNSUPER )
        WRITE ( 6, 1021 ) ( NSUPER(J), J = 1, NSCH )
        WRITE ( 6, 1032 )
 1032   FORMAT ( 1H0, 5X,  4HNBAV )
        WRITE ( 6, 1021 ) ( NBAV(I), I = 1, NOR )
        WRITE ( 6, 1033 )
 1033   FORMAT ( 1H0, 5X,  6HMAXBUS )
        WRITE ( 6, 1021 ) ( MAXBUS(J), J = 1, NSCH )
C       UPDATE K1
  145   K1 = K1 + 1
C       TEST - K1 = NSCH
        IF ( K1 - NSCH ) 149, 149, 1
C       READ DATA FROM TAPE OR DISC A  FOR SCHOOL K1
  149   READ (2) KK1, (LSCH(I), I=1,6 ), N, KS, KSS, KPER, KPASS,
     1 (  ( MOR(I,J), I=1,NOR ), J=1,N ), ( ( M(I,J), I=1,N ), J=1,N ),
     2 ( LAB(I), I=1,N ), ( LOAD(I), I=1,N ), MAXBL, MAXRT
        READ (2) ( LB01(I), I = 1, NOR ), ( LB02(I), I = 1, NOR )
```

```
C     LB1 = LOWER BOUND ON A SET OF ROUTES FOR SCHOOL K1 WHEN MINBUS
C              ROUTES ARE USED AND ALL ROUTES START AT  NSUPER(K1)
C     LB2 - SAME AS LB1 WHEN MAXBUS(K1) ROUTES ARE USED
      I = NSUPER(K1)
      LB1 = LB01(I)
      LB2 = LB02(I)
  163 NP1 = N + 1
      NM1 = N - 1
      NM2 = N - 2
C     MAXTT1 = MAXIMUM TOTAL TIME ALLOWED FOR THE SET OF ROUTES
C                FOR SCHOOL K1  WHEN MINBUS  ROUTES ARE USED
C     MAXTT2 - SAME AS MAXTT1  WHEN MAXBUS(K1)  ROUTES ARE USED
      PER = KPER
      PER = .01 * PER
      XLB1 = LB1
      XLB2 = LB2
      XTT1 = ((1.0 + PER ) * XLB1 ) + 0.999999
      XTT2 = ((1.0 + PER ) * XLB2 ) + 0.999999
      MAXTT1 = XTT1
      MAXTT2 = XTT2
C     NPASS = NUMBER OF CURRENT INFINITE BUS ROUTE BEING GENERATED
C     NOTE - EVERY INFINITE BUS ROUTE CANNOT BE GENERATED INTO A SET
C            OF ROUTES WHICH SATISFY  BUS CAPACITY,  PASSENGER RIDING
C            TIME  AND  BUS AVAILABILITY CONSTRAINTS
      NPASS = 0
C     KODE2 = CODE FOR METHOD OF GENERATING THE INFINITE BUS ROUTE
C     KODE2 = 1  MEANS USE THE NEAREST-CITY METHOD
C     KODE2 = 2  MEANS USE ALGORITHMS
      KODE2 = 1
C     IT1 = TOTAL TRAVELING TIME REQUIRED BY THE BEST INFINITE BUS ROUTE
      IT1 = 77777
C     IR1 = VECTOR CONTAINING THE BEST INFINITE BUS ROUTE
      DO  169  I = 1, N
      IR1(I) = 0
  169 CONTINUE
C      MINTT = TOTAL TRAVELING TIME REQUIRED BY THE BEST SET OF ROUTES
       MINTT = 77777
C      MPASS = NUMBER OF PASS GENERATING THE BEST SET OF ROUTES
       MPASS = 0
C      MINRT = VECTOR CONTAINING THE BEST SET OF ROUTES FROM NSUPER(K1)
C      THE FORMAT OF MINRT IS
C      WORD 1 = NUMBER OF ROUTES IN THE SET
C      FOR EACH ROUTE
C      WORD 1 = NUMBER OF STOPS SERVICED EXCLUDING ORIGIN AND TERMINUS
C      WORDS 2 ... LIST OF STOPS VISITED
       K = N + MAXBUS(K1) - 1
       DO  173  I = 1, K
       MINRT(I) = 0
  173 CONTINUE
C      SHIFT ROW  NSUPER(K1)  FROM MATRIX  MOR  TO ROW  1  OF MATRIX M
       K = NSUPER(K1)
       DO  177  J = 1, N
       M(1,J) = MOR(K,J)
  177 CONTINUE
C      INTERMEDIATE PRINT-OUT - ALL DATA FOR SCHOOL
       IF ( KS ) 182, 182,  56
   56 WRITE ( 6, 1003 )
       WRITE ( 6, 1004 ) ( LSCH(I), I = 1, 6 )
       WRITE ( 6, 1005 )  MAXBL, MAXRT
       WRITE ( 6, 1019 )
 1019 FORMAT ( 1H0, 5X, 10HMATRIX MOR )
       DO 57 I = 1, NOR
       WRITE ( 6, 1020 ) I
       WRITE ( 6, 1021 ) ( MOR(I,J), J = 1, N )
   57 CONTINUE
 1020 FORMAT ( 1H0, 5X, 3HROW, 2X, I4 )
 1021 FORMAT ( 1H , 5X, 20I6 )
       WRITE ( 6, 1022 )
 1022 FORMAT ( 1H0, 5X, 8HMATRIX M )                                   3625
      DO 58 I = 1, N                                                   3626
      WRITE ( 6, 1020 ) I                                              3627
      WRITE ( 6, 1021 ) ( M(I,J), J = 1, N )                           3628
   58 CONTINUE                                                         3629
      WRITE ( 6, 1023 )                                                363
 1023 FORMAT ( 1H0, 5X,  6HLABELS )                                    3631
      WRITE ( 6, 1024 ) ( LAB(I), I = 1, N )                           3632
 1024 FORMAT ( 1H , 5X, 20A6 )                                         3633
      WRITE ( 6, 1025 )                                                3634
 1025 FORMAT ( 1H0, 5X, 5HLOADS )                                      3635
      WRITE ( 6, 1021 ) ( LOAD(I), I = 1, N )                          3636
      WRITE ( 6, 1027 )                                                3639
 1027 FORMAT ( 1H0, 5X, 4HLR01 )                                       364
      WRITE ( 6, 1021 ) ( LR01(I), I = 1,NOR )                         3641
      WRITE ( 6, 1028 )                                                3642
 1028 FORMAT ( 1H0, 5X, 4HLR02 )                                       3643
      WRITE ( 6, 1021 ) ( LR02(I), I= 1,NOR )                          3644
      WRITE ( 6, 1034 ) LB1, LB2, MAXTT1, MAXTT2                       3645
 1034 FORMAT ( 1H0, 5X, 3HLB1, 2X, I6, 4X, 3HLB2, 2X, I6, 4X,          3646
     1 6HMAXTT1, 2X, I6, 4X, 6HMAXTT2, 2X, I6 )                        3647
C     IRN01 = NUMBER OF THE FIRST STOP VISITED BY THE INFINITE BUS ROUTE 365
C             FOR THE NEAREST-CITY METHOD                              3651
  182 IRN01 = 1                                                        3652
C     UPDATE IRN01                                                     3653
  183 IRN01 = IRN01 + 1                                                366
C     TEST- IRN01 = N                                                  367
      IF ( IRN01 - N ) 187, 791, 791                                   368
C     CLEAR  M(I,I), M(I,NP1), M(NP1,I) FOR NEAREST-CITY APPROACH      369
  187 DO  189  I = 1, N                                                370
      M(I,I) = 0                                                       371
      M(I,NP1) = 0                                                     372
      M(NP1,I) = 0                                                     373
  189 CONTINUE                                                         374
```

```
C     IRTP = VECTOR CONTAINING THE SET OF ROUTES OBTAINED FROM THE
C             CURRENT INFINITE BUS ROUTE
C     THE FORMAT OF IRTP IS THE SAME AS THE FORMAT OF MINRT
C     UPDATE NPASS
  191 NPASS = NPASS + 1
      IF ( NPASS - KPASS ) 192, 192, 610
  192 K = N + N + 1
      DO  193  I = 1, K
      IRTP(I) = 0
  193 CONTINUE
C     IRTF = VECTOR CONTAINING THE CURRENT INFINITE BUS ROUTE
      DO 197 I = 1, N
      IRTF(I) = 0
  197 CONTINUE
C     GENERATE THE NEXT INFINITE BUS ROUTE
      GO TO ( 201, 801 ), KODE2
C     THE NEAREST-CITY APPROACH -    IRNO1 IS THE FIRST STOP VISITED
  201 IRNO = IRNO1
      M( 1,NP1 ) = IRNO
      M(NP1,IRNO) = 1
      M(1,1) = M(1,IRNO)
      M(NP1,1) = N
      M(N,NP1) = 1
      DO  213  I = 2, NM2
      MIN = 77777
      IDMIN = 0
      DO  209  J = 2, NM1
      IF ( M(J,NP1) )  209, 203, 209
  203 IF ( IRNO - J )  205, 209, 205
  205 IF ( M( IRNO, J ) - MIN )  207, 209, 209
  207 MIN = M( IRNO, J )
      IDMIN = J
  209 CONTINUE
      M( IRNO, NP1 ) = IDMIN
      M( NP1, IDMIN ) = IRNO
      M( IRNO, IRNO ) = MIN
      IRNO = IDMIN
  213 CONTINUE
      M( IRNO, NP1 ) = N
      M( NP1, N ) = IRNO
      M( IRNO, IRNO ) = M( IRNO, N )
C     ITT = TOTAL TRAVELING TIME FOR THE CURRENT INFINITE BUS ROUTE
C     DETERMINE  ITT  AND  VECTOR  IRTE
  219 ITT = 0
      IRTE(1) = 1
      I = 1
      DO  221  K = 2, N
      J = M( I, NP1 )
      ITT = ITT + M(I,J)
      IRTE(K) = J
      I = J
  221 CONTINUE
C     TEST - ITT = IT1
      IF ( ITT - IT1 ) 223, 230, 230
C     THE CURRENT INFINITE BUS ROUTE IS BEST
  223 IT1 = ITT
      DO  225  I = 1, N
      IR1(I) = IRTE(I)
  225 CONTINUE
C     PARTITIONING INFINITE BUS ROUTE INTO INDIVIDUAL CONSTRAINED ROUTES
C     NR = NUMBER OF CURRENT ROUTE
C     NBS = NUMBER OF STOPS COMPLETELY SERVICED
C     NBS1 = IDENTIFICATION NUMBER OF STOP 1  ON CURRENT ROUTE
C     NBSRT = NUMBER OF BUS STOPS ASSIGNED TO CURRENT ROUTE
C     ITBS1 = TOTAL RIDING TIME FROM BUS STOP 1  ON CURRENT ROUTE
C     NBSOLD = NUMBER OF PREVIOUS BUS STOP VISITED ON CURRENT ROUTE
C     NBSNEW = NUMBER OF PRESENT BUS STOP VISITED ON CURRENT ROUTE
C     IAROLD = ARRIVAL TIME AT PREVIOUS BUS STOP
```

POOR ORIGINAL COPY - BEST
AVAILABLE AT TIME FILMED

```
C     IARNEW = ARRIVAL TIME AT PRESENT BUS STOP
C     IBLOLD = BUS LOAD AT PREVIOUS STOP
C     IBLNEW = BUS LOAD AT PRESENT STOP
C     KODE3 = 0  MEANS THE CURRENT ROUTE IS INCOMPLETE
C     KODE3 = 1  MEANS THE CURRENT ROUTE IS COMPLETE
C     KODE3 = 2  MEANS ALL THE BUS STOPS HAVE BEEN SERVICED
  230 NR = 0
      NBS = 0
C     INITIALIZE FOR A NEW ROUTE
  234 NR = NR + 1
      KODE3 = 0
      NBSOLD = 0
      NBSNEW = 1
      IAROLD = 0
      IARNEW = 0
      IBLOLD = 0
      IBLNEW = 0
      ITBS1 = 0
      NBSRT = 0
      NBS1 = IRTE( NBS + 2 )
  248 NBSOLD = NBSNEW
      IAROLD = IARNEW
      IBLOLD = IBLNEW
      NBSNEW = IRTE( NBS + 2 )
      IARNEW = IAROLD + M( NBSOLD, NBSNEW )
      IBLNEW = IBLOLD + LOAD( NBSNEW )
      ITBS1 = IARNEW + M( NBSNEW, N ) - M( 1, NBS1 )
C     TEST - BUS CAPACITY EXCEEDED
  254 IF ( IBLNEW - MAXBL ) 256, 266, 270
C     AT 256 THE BUS IS NOT FULL
C     TEST - RIDING TIME EXCEEDED
  256 IF ( ITBS1 - MAXRT ) 273, 258, 270
C     AT 258 THE BUS IS NOT FULL BUT MAXIMUM RIDING TIME REACHED
C     CURRENT ROUTE IS COMPLETE
```

```
  258 KODE3 = 1
      GO TO 273
C     AT 266 THE BUS IS FULL
C     TEST - RIDING TIME EXCEEDED
  266 IF ( ITBS1 - MAXRT ) 268, 268, 270
C     AT 268 THE BUS IS FULL BUT THE RIDING TIME HAS NOT BEEN EXCEEDED
C     CURRENT ROUTE IS COMPLETE
  268 KODE3 = 1
      GO TO 273
C     AT 270 SERVICING THE CURRENT STOP WOULD VIOLATE A CONSTRAINT
C     THE PREVIOUS STOP COMPLETED THE ROUTE
  270 KODE3 = 1
      GO TO 275
C     AT 273 THE CURRENT STOP WILL BE ASSIGNED TO THE CURRENT ROUTE
  273 NBS = NBS + 1
      NBSRT = NBSRT + 1
      K = NR + NBS + 1
      IRTP( K ) = NBSNEW
C     TEST - ALL STOPS SERVICED
  275 IF ( NBS - N + 2 )  279, 277, 277
C     AT 277 ALL STOPS SERVICED
  277 KODE3 = 2
C     TEST - PARTITIONING COMPLETED
  279 IF ( KODE3 - 1 ) 248, 281, 281
C     AT 280 THE CURRENT ROUTE IS COMPLETE BECAUSE EITHER THE CONSTRAINT
C     LIMITS WERE REACHED OR ALL THE STOPS WERE SERVICED
C     TEST - AT LEAST ONE STOP ON ROUTE
  281 IF ( NBSRT ) 285, 285, 283
C     AT 283 THE CURRENT ROUTE IS LEGAL, STORE NBSRT IN VECTOR IRTP
  283 K = NR + NBS - NBSRT + 1
      IRTP( K ) = NBSRT
      IF ( KODE3 - 1 ) 234, 234, 287
C     AT 285 THE CURRENT ROUTE IS NOT LEGAL, GENERATE NEXT INFINITE RTE.
  285 GO TO ( 183, 191 ), KODE2
```

```
C     AT 287 PARTITIONING IS COMPLETE
  287 IRTP(1) = NR
C     INTERMEDIATE PRINT-OUT
      IF ( KS ) 293, 293, 289
  289 WRITE ( 6, 1035 ) NPASS, KPASS, KODE2, IT1, ITT, MINTT, MPASS
 1035 FORMAT ( 1H0, 5X, 5HNPASS, I6, 5X, 5HKPASS, I6, 5X, 5HKODE2,
     1 I6, 5X, 3HIT1, I6, 5X, 3HITT, I6, 5X, 5HMINTT, I6, 5X,
     2 5HMPASS, I6 )
      WRITE ( 6, 1036 )
 1036 FORMAT ( 1H0, 5X, 10HVECTOR IR1  )
      WRITE ( 6, 1021 ) ( IR1(I), I = 1, N )
      WRITE ( 6, 1037 )
 1037 FORMAT ( 1H0, 5X, 10HVECTR IRTE )
      WRITE ( 6, 1021 ) ( IRTE(I), I = 1, N )
      WRITE ( 6, 1038 )
 1038 FORMAT ( 1H0, 5X, 10HVECTR IRTP )
      K = NR + N - 1
      WRITE ( 6, 1021 ) ( IRTP(I), I = 1, K )
      WRITE ( 6, 1045 )
 1045 FORMAT ( 1H0, 5X, 5HMINRT )
      K = N + MAXBUS(K1) - 1
      WRITE ( 6, 1021 ) ( MINRT(I), I = 1, K )
C     TEST - NR SATISFY BUS AVAILABILITY CONSTRAINT
  293 IF ( NR - MAXBUS(K1) )  305, 305, 295
  295 GO TO ( 183, 191 ), KODE2
C     AT 305  NR  DOES NOT EXCEED MAXBUS(K1)
C     IMPROVE EACH OF THE INDIVIDUAL ROUTES
C     ITIMP = TOTAL TRAVELING TIME REQUIRED BY THE SET OF IMPROVED ROUTE
  305 NR = 0
      NRS = 0
      ITIMP = 0
C     INITIALIZE FOR IMPROVING CURRENT ROUTE
  309 NR = NR + 1
C     TEST - ALL ROUTES OF VECTOR IRTP  IMPROVED
```

```
      IF ( NR - IRTP(1) )  313, 313, 575
C     AT 313 AT LEAST ONE ROUTE HAS TO BE IMPROVED
C     NRSRT = NUMBER OF STOPS ON THE CURRENT ROUTE EXCLUDING THE
C                   ORIGIN AND THE TERMINUS
  313 K = NR + NBS + 1
      NRSRT = IRTP( K )
C     TEST - NBSRT = 1
      IF ( NRSRT - 1 )   317, 317, 321
C     AT 317 THE CURRENT ROUTE HAS ONE STOP. NO IMPROVEMENT POSSIBLE
  317 NRS = NBS + NBSRT
      J = IRTP( K+1 )
      ITIMP = ITIMP + M(1,J) + M(J,N)
      GO TO 309
C     KDE(I) = 0 MEANS ROW I  AND COLUMN I  OF MATRIX M  WILL NOT
C                 BE USED FOR THE IMPROVEMENT PROCESS FOR ROUTE K2
C     KDE(I) = 1 MEANS ROW I  AND COLUMN I  OF MATRIX M  WILL
C                 BE USED FOR THE IMPROVEMENT PROCESS FOR ROUTE K2
  321 DO  325  I = 2, NM1
      KDE(I) = 0
  325 CONTINUE
C     ROW 1, ROW N, COL.1, COL.N  OF MATRIX M  ARE ALWAYS USED
      KDE(1) = 1
      KDE(N) = 1
C     IRTF2 = VECTOR CONTAINING THE CURRENT ROUTE EXPRESSED AS AN
C                INFINITE BUS ROUTE- INCLUDES ORIGIN AND TERMINUS
C     N2 = NUMBER OF STOPS ON THE ROUTE BEING IMPROVED
C     N2 INCLUDES THE ORIGIN AND THE TERMINUS
  329 N2 = NBSRT + 2
      DO 330 I = 1, N2
      IRTF2(I) = 0
  330 CONTINUE
      J = NR + NBS + 1
      DO  333  I = 1, NBSRT
      K = J + I
      IRS = IRTP( K )
      IRTF2( I+1 ) = IRS
      KDE( IRS ) = 1
  333 CONTINUE
      IRTF2(1) = 1
      IRTF2(N2) = N
C     NOTE - ALL NUMBERING OF STOPS IS WITH RESPECT TO MATRIX M IN IRTF2
C     MATRIX M2  IS THE TIME MATRIX FOR THE INDIVIDUAL ROUTE
C     TO BE IMPROVED.  IT IS OF ORDER  N2
C     I1 = ROW NUMBER OF MATRIX M  TO BE SHIFTED TO MATRIX M2
C     J1 = COLUMN NUMBER OF MATRIX M  TO BE SHIFTED TO MATRIX M2
C     LIST = VECTOR OF ROW-COLUMN NUMBERS DESIGNATING THE
C             CONSTRUCTION OF MATRIX M2  FROM MATRIX M
C     K = COUNTER FOR POSITION IN VECTOR LIST
  337 K = 0
      DO  345  I = 1, N
      IF ( KDE(I) )  341, 345, 341
C     AT 341 ROW I  OF MATRIX M  IS TO BE SHIFTED TO ROW K  OF MATRIX M2
  341 K = K + 1
      LIST(K) = I
  345 CONTINUE
      DO  353  I = 1, N2
      I1 = LIST(I)
      DO  349  J = 1, N2
      J1 = LIST(J)
      M2( I,J ) = M( I1,J1 )
  349 CONTINUE
  353 CONTINUE
C     RELATE EACH ROW OF MATRIX M2  TO VECTOR IRTF2 SO THAT THE ROUTE
C     BEING IMPROVED IS THE INITIAL ROUTE FOR ALGORITHMS A  AND B
      N2P1 = N2 + 1
      N2M1 = N2 - 1
      M2( N2, N2P1 ) = 1
      M2( N2P1, 1 ) = N2
      M2( N2P1, N2P1 ) = 0
C     I1 = COUNTER FOR PICKING OFF THE STOPS FROM  IRTE2
C     J1 AND KK1 ARE THE STOP NUMBERS WITH RESPECT TO MATRIX M
C     J AND K  ARE THE STOP NUMBERS WITH RESPECT TO MATRIX M2
      DO  379  I1 = 1, N2M1
C     J1 = TERMINUS OF CURRENT LINK, NUMBERING W.R.T. MATRIX M
      J1 = IRTE2( I1 + 1 )
C     KK1 = ORIGIN OF CURRENT LINK, NUMBERING W.R.T. MATRIX M
      KK1= IRTE2( I1 )
      DO  365  I2 = 1, N2
      IF ( LIST(I2) - J1 ) 365, 361, 365
C     AT 361 STOP J1  OF MATRIX  M  IS STOP I2  OF MATRIX M2
  361 J = I2
      GO TO 367
  365 CONTINUE
  367 DO  371  I2 = 1, N2
      IF ( LIST(I2) - KK1) 371, 369, 371
C     AT 369 STOP KK1 OF MATRIX  M  IS STOP I2  OF MATRIX M2
  369 K = I2
      GO TO 375
  371 CONTINUE
  375 M2( K, N2P1 ) = J
      M2( N2P1, J ) = K
      M2(K,K) = M2(K,J)
  379 CONTINUE
C     MATRIX M2(I,J) IS READY FOR ALGORITHM A
C     ALGORITHM A
  420 DO  460  I = 1, N2M1
      DO  450  J = 2, N2M1
      IF ( I - J ) 422, 450, 422
  422 IF ( M2( I, N2P1 ) - J ) 424, 450, 424
  424 K01 = I
      KT1 = J
      K02 = J
```

POOR ORIGINAL COPY - BEST AVAILABLE AT TIME FILMED

```
      KT2 = M2( I, N2P1 )
      KO3 = M2( N2P1, J )
      KT3 = M2( J, N2P1 )
  426 KOLD = M2( KO1, KO1 ) + M2( KO2, KO2 ) + M2( KO3, KO3 )
      NEW = M2( KO1, KT1 ) + M2( KO2, KT2 ) + M2( KO3, KT3 )
      IF ( NEW - KOLD ) 470, 427, 427
  427 IF ( KO1 - KT3 )  428, 450, 428
  428 IF ( KT3 - N2 ) 430, 429, 430
  429 IF ( KO1 - 1 ) 431, 450, 431
  431 KO2 = 1
      KT3 = M2( 1, N2P1 )
      GO TO 426
  430 KO2 = KT3
      KT3 = M2( KO2, N2P1 )
      GO TO 426
  450 CONTINUE
  460 CONTINUE
C     AT THIS POINT ALGORITHM A  IS EXHAUSTED
  462 GO TO 550
  470 M2( KO1, KO1 ) = M2( KO1, KT1 )
      M2( KO2, KO2 ) = M2( KO2, KT2 )
      M2( KO3, KO3 ) = M2( KO3, KT3 )
      M2( KO1, N2P1 ) = KT1
      M2( KO2, N2P1 ) = KT2
      M2( KO3, N2P1 ) = KT3
      M2( N2P1, KT1 ) = KO1
      M2( N2P1, KT2 ) = KO2
      M2( N2P1, KT3 ) = KO3
      GO TO 420
C     AT THIS POINT THE CURRENT ROUTE HAS BEEN IMPROVED
C     ALL ROUTE DATA IS STORED IN MATRIX M2
C     IRTT = TOTAL TIME REQUIRED TO TRAVERSE THE IMPROVED ROUTE
C     IRTF3 = VECTOR CONTAINING THE CURRENT IMPROVED ROUTE EXPRESSED AS
C             AN INFINITE BUS ROUTE- INCLUDES ORIGIN AND TERMINUS
```

```
C       NOTE- ALL NUMBERING OF STOPS IS WITH RESPECT TO MATRIX M2 IN IRTE3
   550 IRTT = 0
       DO 551 I = 1, N2
       IRTE3(I) = 0
   551 CONTINUE
       IRTE3(1) = 1
C      I = ORIGIN OF CURRENT LINK, NUMBERING W.R.T. MATRIX  M2
       I = 1
       DO  553  K = 2, N2
C      J = TERMINUS OF CURRENT LINK, NUMBERING W.R.T. MATRIX M2
       J = M2( I, N2P1 )
       IRTT = IRTT + M2( I,J )
       IRTE3( K ) = J
       I = J
   553 CONTINUE
C      UPDATE  ITIMP
       ITIMP = ITIMP + IRTT
C      RELATE THE BUS STOP NUMBERS OF VECTOR IRTE3 IN TERMS OF THE
C      BUS STOP NUMBERS OF MATRIX M  AND VECTOR IRTP
C      STORE THE IMPROVED ROUTE BACK PROPERLY IN VECTOR IRTP
       K = 0
       L = NR + NBS + 1
       DO  561  I = 2, N2M1
C      THE ORIGIN AND TERMINUS IS NOT STORED IN VECTOR IRTP
       K = K + 1
       J = IRTE3( I )
       L1= L + K
       IRTP(L1 ) = LIST( J )
   561 CONTINUE
C      UPDATE NBS
       NBS = NBS + NBSRT
C      IMPROVE NEXT ROUTE OF SET
       GO TO 309
C      AT 575 ALL THE ROUTES OF VECTOR IRTP HAVE BEEN IMPROVED
```

```
C      INTERMEDIATE PRINT-OUT
  575 IF ( KS ) 578, 578, 576
  576 WRITE ( 6, 1046 ) ITIMP
 1046 FORMAT ( 1H0, 5X, 5HITIMP, I6 )
      K = IRTP(1) + N - 1
      WRITE ( 6, 1038 )
      WRITE ( 6, 1021 ) ( IRTP(I), I = 1, K )
C     TEST - ITIMP LESS THAN MINTT
  578 IF ( ITIMP - MINTT ) 583, 579, 579
C     AT 579 GENERATE ANOTHER INFINITE BUS ROUTE
  579 GO TO ( 183, 191 ), KODE2
C     AT 583 THE CURRENT SET OF ROUTES IS THE BEST AVAILABLE
  583 MINTT = ITIMP
      MPASS = NPASS
      NR = IRTP(1)
      K = NR + N - 1
      DO  585  I = 1, K
      MINRT(I) = IRTP(I)
  585 CONTINUE
C     TEST - NR = MAXBUS(K1)
      IF ( NR - MAXBUS(K1) )  600, 603, 603
C     TEST - CURRENT SET OF ROUTES ACCEPTABLE AS FINAL SOLUTION
  600 IF ( MINTT - MAXTT1 ) 610, 610, 579
  603 IF ( MINTT - MAXTT2 ) 610, 610, 579
C     AT 610 A FINAL ACCEPTABLE SOLUTION MAY EXIST
  610 IF ( MINRT(1) ) 604, 604, 606
  604 WRITE ( 6, 1003 )
      WRITE ( 6, 999 )
      WRITE ( 6, 1004 ) ( LSCH(I), I = 1,6 )
      WRITE ( 6, 1047 )
 1047 FORMAT ( 1H0, 5X, 42HNO ACCEPTABLE SOLUTION. ADJUST CONSTRAINTS )
      GO TO 145
C     AT 606 THE BEST SET OF ROUTES IS ACCEPTABLE AS A FINAL SOLUTION
C     LIST(J) = IDENTIFICATION NUMBER OF ORIGIN J SUPPLYING BUSES
```

```
C                 FOR SCHOOL K1 - DETERMINED AT STATEMENT 129
C        CLEAR VECTOR LIST(J)
  606 DO  611  J = 1, NOR
      LIST(J) = 0
  611 CONTINUE
C     KODE4 = CODE FOR CALCULATING FINAL LOWER BOUND
      KODE4 = 1
C     TEST - MORE THAN ONE ORIGIN
      IF ( NOR - 1 ) 775, 775, 612
C     TEST - ORIGIN NSUPER(K1) SUPPLIES ALL THE REQUIRED BUSES
  612 I = NSUPER(K1)
      IF ( NBUS( I,K1) - MAXBUS(K1) ) 613, 875, 875
C     DEVELOP COST MATRIX FOR ALLOCATION OF ORIGINS TO BUS ROUTES
C     KONRT(I) = CONSTANT PORTION OF ROUTE COST FOR ROUTE I
C     KONRT(I) = SUM OF ALL LINKS EXCEPT THE ONE FROM ORIGIN TO STOP 1
C     NSTOP1(I) = IDENTIFICATION NUMBER OF STOP 1  ON ROUTE I
  613 NR = 0
      NBS = 0
  617 NR = NR + 1
C     TEST - ALL ROUTES UNPACKED
      IF ( NR - MINRT(1) ) 621, 621, 641
  621 L1 = NR + NBS + 1
      L2 = L1 + 1
      NBSRT = MINRT( L1 )
      KONRT( NR ) = 0
      K = NBSRT - 1
      IF ( K ) 633, 633, 625
  625 DO  629  I = 1, K
      I1 = L1 + I
      J1 = L2 + I
      IBS1 = MINRT( I1 )
      IBS2 = MINRT( J1 )
      KONRT( NR ) = KONRT( NR ) + M( IBS1, IBS2 )
  629 CONTINUE
  633 L3 = L1 + NBSRT
      IBS1 = MINRT( L3 )
      IBS2 = N
      KONRT( NR ) = KONRT( NR ) + M( IBS1, IBS2 )
      NSTOP1( NR ) = MINRT( L2 )
C     UPDATE NBS
      NBS = NBS + NBSRT
      GO TO 617
C     KOST(I,J) = TOTAL TRAVELING TIME FOR ROUTE J  STARTING AT ORIGIN I
C     NOR2 = NUMBER OF ORIGINS WHICH SUPPLY BUSES FOR SCHOOL K1
C     IREQ(J) = NUMBER OF BUSES REQUIRED BY ROUTE  J
  641 NR = MINRT(1)
      NOR2 = 0
      DO  649  I = 1, NOR
      IF ( NBUS( I,K1 ) ) 649, 649, 643
C     AT 643 ORIGIN I  SUPPLIES AT LEAST ONE BUS FOR SCHOOL K1
C     UPDATE NOR2, VECTORS LIST AND IVAIL
C     IVAIL(I) = NUMBER OF BUSES AVAILABLE AT ORIGIN I  FOR SCHOOL K1
  643 NOR2 = NOR2 + 1
      LIST(NOR2) = I
      IVAIL(NOR2) = NBUS(I,K1)
      DO  645  J = 1, NR
      K = NSTOP1(J)
      KOST( NOR2,J ) = KONRT(J) + MOR( I,K )
  645 CONTINUE
  649 CONTINUE
      DO  653  J = 1, NR
      IREQ(J) = 1
  653 CONTINUE
C     EXECUTE THE TRANSPORTATION ALGORITHM
      CALL TRAN ( NOR2,NR,IVAIL,IREQ,KOST,NBUS1,KODE )
      IF ( KS ) 713, 713, 650
  650 WRITE ( 6, 1051 )
 1051 FORMAT ( 1H0, 5X, 4HKOST )
```

```
      DO  651  I = 1, NOR2
      WRITE ( 6, 1021 ) ( KOST(I,J), J = 1, NR )
  651 CONTINUE
      WRITE ( 6, 1052 )
 1052 FORMAT ( 1H0, 5X, 5HNBUS1 )
      DO  652  I = 1, NOR2
      WRITE ( 6, 1021 ) ( NBUS1(I,J), J = 1, NR )
  652 CONTINUE
      WRITE ( 6, 1053 )
 1053 FORMAT ( 1H0, 5X, 5HIVAIL )
      WRITE ( 6, 1021 ) ( IVAIL(I), I = 1, NOR2 )
      WRITE ( 6, 1054 )
 1054 FORMAT ( 1H0, 5X, 4HIREQ )
      WRITE ( 6, 1021 ) ( IREQ(I), I = 1, NR )
      WRITE ( 6, 1055 )
 1055 FORMAT ( 1H0, 5X, 4HLIST )
      WRITE ( 6, 1021 ) ( LIST(I), I = 1, NOR2 )
      GO TO 713
C     CALCULATE THE FINAL LOWER BOUND FOR THE ROUTES OF SCHOOL K1
C     NBUS1(I,J) = 0  MEANS ORIGIN I  DOES NOT SERVICE ROUTE J
C     NBUS1(I,J) = 1  MEANS ORIGIN I  SERVICES ROUTE J
C     NBUS2(I) = NUMBER OF ROUTES STARTING FROM ORIGIN I
C     NBTOT = TOTAL NUMBER OF BUSES USED BY SCHOOL K1
  660 NR = MINRT(1)
      DO  665  I = 1, NOR2
      NBUS2(I) = 0
      DO  661  J = 1, NR
      NBUS2(I) = NBUS2(I) + NBUS1( I,J )
  661 CONTINUE
  665 CONTINUE
C     LB = FINAL LOWER BOUND FOR SCHOOL K1
      LB = LBS
C     FIND PORTION OF LOWER BOUND CONTRIBUTED BY TRAVELING BETWEEN
C     ORIGIN I  AND THE FIRST STOP OF ANY ROUTE
```

```
      DO  681  I = 1, NOR2
      K = LIST(I)
      MIN = 77777
      DO  673  J = 2, NM1
      IF ( MOR(K,J) - MIN ) 669, 673, 673
  669 NCOL = J
      MIN = MOR(K,J)
  673 CONTINUE
      KDE(NCOL) = 1
      LB = LB + ( NBUS2(I) * MIN )
C     REDUCE ROW K  OF MATRIX MOR  BY  MIN
      DO  677  J = 2, NM1
      MOR(K,J) = MOR(K,J) - MIN
  677 CONTINUE
  681 CONTINUE
C     FIND PORTION OF LOWER BOUND CONTRIBUTED BY COLUMNS
      DO  705  J = 2, N
      IF ( KDE(J) ) 685, 685, 705
  685 MIN = 77777
      DO  693  I = 1, NOR2
      K = LIST(I)
      IF ( MOR(K,J) - MIN ) 689, 693, 693
  689 MIN = MOR(K,J)
  693 CONTINUE
      DO  701  I = 2, NM1
      IF ( M(I,J) - MIN ) 697, 701, 701
  697 MIN = M(I,J)
  701 CONTINUE
      IF ( J - N ) 702, 703, 703
  702 LB = LB + MIN
      GO TO 705
  703 LB = LB + ( NR * MIN )
  705 CONTINUE
      GO TO 773
```

```
C     WRITE THE CURRENT ROUTE
C     NR = NUMBER OF ROUTE BEING PRINTED
C     NBS = NUMBER OF BUS STOPS OF VECTOR MINRT PRINTED
  713 NR = 0
      NBS = 0
C     INITIALIZE FOR CURRENT ROUTE
  715 NR = NR + 1
C     TEST- ALL ROUTES PRINTED
      IF ( NR - MINRT(1) ) 719, 719, 767
  719 K3 = NR + NBS
      NBSRT = MINRT( K3 + 1 )
C     NBSRT = NUMBER OF STOPS ON CURRENT ROUTE - EXCLUDING ORG. AND TER.
      K = NBSRT + 1
C     ILOAD(I) = TOTAL BUS LOAD AT STOP I
C     IARRT(I) = ARRIVAL TIME AT STOP I
C     INUM(I) = NUMERIC LABEL FOR STOP I
C     IALPH(I) = ALPHAMERIC LABEL FOR STOP I
      DO  731  I = 1, NOR2
      IF ( NBUS1( I, NR ) ) 731, 731, 727
C     AT 727 ORIGIN I  SERVES ROUTE NR
  727 K4 = LIST(I)
      GO TO 735
  731 CONTINUE
  735 DO  751  I = 1, K
      IF ( I-2 ) 739, 743, 747
  739 INUM(I) = I
      IALPH(I) = LOR(K4)
      IARRT(I) =0
      ILOAD(I) = 0
      GO TO 751
  743 K7 = K3 + I
      K6 = MINRT( K7 )
      INUM(I) = K6
      IALPH(I) = LAB( K6 )
      IARRT(I) = IARRT( I-1 ) + MOR( K4, K6 )
      ILOAD(I) = ILOAD( I-1 ) + LOAD( K6 )
      GO TO 751
  747 K7 = K3 + I
      K6 = MINPT( K7 )
      K5 = MINRT( K7 - 1 )
      INUM(I) = K6
      IALPH(I) = LAB(K6)
      IARRT(I) = IARRT( I-1 ) + M( K5, K6 )
      ILOAD(I) = ILOAD( I-1 ) + LOAD( K6 )
  751 CONTINUE
      K7 = K3 + K
      K6 = N
      K5 = MINRT( K7 )
      I = K + 1
      INUM(I) = K6
      IALPH(I) = LAB(K6)
      IARRT(I) = IARRT( I - 1 ) + M( K5, K6 )
      ILOAD(I) = ILOAD( I - 1 )
      WRITE ( 6, 1003 )
 1003 FORMAT ( 1H1 )
      WRITE ( 6, 999 )
      WRITE ( 6, 1004 ) ( LSCH(J), J= 1, 6 )
 1004 FORMAT ( 1H0, 5X, 6A10 )
      WRITE ( 6, 1005 ) MAXRL, MAXRT
 1005 FORMAT ( 1H0, 5X, 16HMAXIMUM BUS LOAD, I6, 7X,
     1 19HMAXIMUM RIDING TIME, I6 )
      WRITE ( 6, 1050 ) MPASS
 1050 FORMAT ( 1H0, 5X, 37HACCEPTED SET OF ROUTES FROM ITERATION, I6 )
      WRITE ( 6, 1006 ) NR, K4, LOR(K4)
 1006 FORMAT ( 1H0, 5X, 5HROUTE, I5, 22X, 6HORIGIN, I5, 2X, A6 )
      WRITE ( 6, 1007 )
 1007 FORMAT ( 1H0, 5X, 19HSTOP IDENTIFICATION, 6X, 4HTIME, 6X, 4HLOAD )
      WRITE ( 6, 1008 ) (INUM(J), IALPH(J), IARRT(J), ILOAD(J), J=1,I )
 1008 FORMAT ( 1H , 5X, I6, 7X, A6, 6X, I4, 6X, I4 )
C     PREPARE DATA FOR FINAL STATISTICS
C     KRT(I) = NUMBER OF ROUTE I
C     KORN(I) = NUMERIC LABEL FOR ORIGIN OF ROUTE I
C     KORA(I) = ALPHAMERIC LABEL FOR ORIGIN OF ROUTE I
C     KT(I) = TOTAL TIME REQUIRED BY ROUTE I
C     KL(I) = TOTAL BUS LOAD FOR ROUTE I
      KRT(NR) = NR
      KORN(NR) = K4
      KORA(NR) = LOR(K4)
      KT(NR) = IARRT(I)
      KL(NR) = ILOAD(I)
C     UPDATE NBS
      NBS = NBS + NBSRT
      GO TO 715
C     AT 767 WRITE FINAL STATISTICS FOR SCHOOL K1
  767 WRITE ( 6, 1003 )
      WRITE ( 6, 999 )
      WRITE ( 6, 1004 ) ( LSCH(I), I = 1, 6 )
      WRITE ( 6, 1005 ) MAXRL, MAXRT
      WRITE ( 6, 1009 )
 1009 FORMAT ( 1H0, 5X, 5HROUTE, 6X, 15HORIGIN OF ROUTE, 6X, 4HTIME,
     1 6X, 4HLOAD )
      K = MINRT(1)
      WRITE ( 6, 1010 ) (KRT(J), KORN(J), KORA(J), KT(J), KL(J), J= 1,K)
 1010 FORMAT ( 1H0, 5X, I5, 6X, I4, 5X, A6, 6X, I4, 6X, I4 )
      KTOT = 0
      DO  771  J = 1, K
      KTOT = KTOT + KT(J)
  771 CONTINUE
      WRITE ( 6, 1011 ) KTOT
 1011 FORMAT ( 1H0, 5X, 28HTOTAL TIME FOR SET OF ROUTES, I8 )
      READ (2) LBS, ( KDE(I), I=1,N ), ( ( M(I,J), I=1,N ), J=1,N )
      GO TO ( 660, 773 ), KODE4
```

POOR ORIGINAL COPY - BEST
AVAILABLE AT TIME FILMED

```
  773 WRITE ( 6, 1012 ) LB
 1012 FORMAT ( 1H0, 5X, 29HLOWER BOUND FOR SET OF ROUTES, I7 )
      AKTOT = KTOT
      ALB = LB
      RATIO1 = AKTOT / ALB
      PER = KPER
      RATIO2 = (  PER  /100.0 ) + 1.0
      WRITE ( 6, 1013 ) RATIO1
 1013 FORMAT ( 1H0, 21X, 12HACTUAL RATIO, 6X, F5.2 )
      WRITE ( 6, 1014 ) RATIO2
 1014 FORMAT ( 1H0, 21X, 13HALLOWED RATIO, 5X, F5.2 )
      GO TO 145
C     AT 775 THERE IS ONLY 1 ORIGIN - NO NEED TO ASSIGN ORIGINS
C     NBUS1(I,J ) = 0 MEANS ORIGIN I  DOES NOT SERVICE ROUTE J
C     NBUS1(I,J ) = 1 MEANS ORIGIN I  SERVICES ROUTE J
  775 LIST(1) = 1
      I = 1
  777 NR = MINRT(1)
C     NCR2 = NUMBER OF ORIGINS WHICH SUPPLY BUSES FOR SCHOOL K1
      NCR2 = 1
      DO  779  J = 1, NR
      NBUS1( 1,J ) = 1
  779 CONTINUE
      KODE4 = 2
C****               AT THIS POINT SET     KODE4 = 1                    ****
C****                  IF   MAXBUS(K1) = MINBUS + KBUS                 ****
C****     IN ORDER TO CALCULATE CORRECT LOWER BOUND                    ***
C**    **    **    **    **    **    **    **    **    **    **        **
      IF (NR - MAXBUS(K1)   ) 783, 787, 787
C     LB = FINAL LOWER BOUND FOR A SET OF ROUTES
  783 LB = LB01(I)
      GO  TO  713
  787 LB = LB02(I)
      GO  TO  713
```

```
C     AT 791 THE NEAREST-CITY APPROACH HAS BEEN EXHAUSTED
  791 KODF2 = 2
C     STORE THE BEST INFINITE BUS ROUTE INTO MATRIX  M
C     VECTOR  IR1  CONTAINS THE BEST ROUTE CALCULATED
      M( N, NP1 ) = 1
      M( NP1, 1 ) = N
      DO  795  I = 1, NM1
      J = IP1( I+1 )
      K = IR1( I )
      M( K,K ) = M( K,J )
      M( K,NP1) = J
      M( NP1,J) = K
  795 CONTINUE
C     GENERATE THE NEXT INFINITE BUS ROUTE BY ALGORITHMS
      GO TO 191
C     AT 801 GENERATE THE INFINITE BUS ROUTE BY USING THE ALGORITHMS
C     ALGORITHM  A
  801 DO  860  I = 1, NM1
      DO  850  J = 2, NM1
      IF ( I - J ) 822, 850, 822
  822 IF ( M( I,NP1 ) - J ) 824, 850, 824
  824 KO1 = I
      KT1 = J
      KO2 = J
      KT2 = M( I, NP1 )
      KO3 = M( NP1, J )
      KT3 = M( J, NP1 )
  826 KOLD = M( KO1, KO1 ) + M( KO2, KO2 ) + M( KO3, KO3 )
      NEW = M( KO1, KT1 ) + M( KO2, KT2 ) + M( KO3, KT3 )
      IF ( NEW - KOLD ) 870, 827, 827
  827 IF ( KO1 - KT3 ) 828, 850, 828
  828 IF ( KT3 - N ) 830, 829, 830
  829 IF ( KO1 - 1 ) 831, 850, 831
  831 KO2 = 1
```

```
      KT3 = M( 1, NP1 )
      GO TO 826
  830 KO2 = KT3
      KT3 = M( KO2, NP1 )
      GO TO 826
  850 CONTINUE
  860 CONTINUE
C     AT THIS POINT ALGORITHM A  IS EXHAUSTED
C                   WRITE THE BEST SET OF ROUTES AS THE FINAL SOLUTION
      GO TO 610
C     AT 870 AN IMPROVEMENT IN THE INFINITE BUS ROUTE IS POSSIBLE
  870 M( KO1, KO1 ) = M( KO1, KT1 )
      M( KO2, KO2 ) = M( KO2, KT2 )
      M( KO3, KO3 ) = M( KO3, KT3 )
      M( KO1, NP1 ) = KT1
      M( KO2, NP1 ) = KT2
      M( KO3, NP1 ) = KT3
      M( NP1, KT1 ) = KO1
      M( NP1, KT2 ) = KO2
      M( NP1, KT3 ) = KO3
C     PARTITION THE IMPROVED INFINITE BUS ROUTE
      GO TO 219
C     AT 875 ORIGIN NSUPER(K1) SUPPLIES ALL THE REQUIRED BUSES
C     THERE IS ONLY ONE ORIGIN - NO NEED TO ASSIGN ORIGINS TO ROUTES
  875 LIST(1) = NSUPER(K1)
      I = LIST(1)
      GO TO 777
      END

      SUBROUTINE TRAN  ( NORIG, NDEST, IORIG, IDEST, ICOST, IBAS, KE )
      DIMENSION  IORIG(10), IDEST(35), ICOST(10,35), IBAS(10,35)
      DIMENSION  IU(10), IV(35), IU1(10), IV1(35)
      DIMENSION  INET(88), INET1(10), INET2(35)
```

```
C     ARRAYS MUST BE DIMENSIONED AS FOLLOWS -                                   TRAN005
C     LET  I = NUMBER OF ORIGINS                                                TRAN006
C          J = NUMBER OF DESTINATIONS                                           TRAN0061
C     IBAS(I+1, J+1), ICOST(I+1, J+1), IU(I+1), IV(J+1), IU1(I+1)               TRAN0062
C     IV1(J+1), INET( 2I + 2J + 2 ), INET1(I+1), INET2(J+1)                     TRAN0063
C     IORIG(I+1), IDEST(J+1)                                                    TRAN0064
C**** IORIG(I)  MUST BE GREATER THAN ZERO FOR ALL  I                         ****TRAN0065
C**** IDEST(J)   MUST BE GREATER THAN ZERO FOR ALL  J                        ****TRAN0066
C**    **   **   **   **   **   **   **   **   **   **                          TRAN0067
C     TRANSPORTATION ALGORITHM, MODI METHOD                                     TRAN007
C     K1 AND K2 ARE CODES AND CONSTANTS FOR COMPARISON                          TRAN008
 1121 K1 = 10000000                                                             TRAN009
      K2 = 100000000                                                            TRAN010
C     NORIG = NUMBER OF ORIGINS                                                 TRAN011
C     NDEST = NUMBER OF DESTINATIONS                                            TRAN012
      NDEST1 = NDEST                                                            TRAN013
      NORIG1 = NORIG                                                            TRAN014
C     ISUMO = TOTAL NUMBER UNITS AVAILABLE                                      TRAN015
C     ISUMD = TOTAL NUMBER UNITS REQUIRED                                       TRAN016
 1171 ISUMO = 0                                                                 TRAN017
      ISUMD = 0                                                                 TRAN018
      DO 1102  I = 1, NORIG                                                     TRAN019
C     IORIG(I) = NUMBER OF UNITS AVAILABLE AT ORIGIN I                          TRAN020
      ISUMO = ISUMO + IORIG( I )                                                TRAN021
 1102 CONTINUE                                                                  TRAN022
      DO 1103  J = 1, NDEST                                                     TRAN023
C     IDEST(J) = NUMBER OF UNITS REQUIRED AT DESTINATION J                      TRAN024
      ISUMD = ISUMD + IDEST( J )                                                TRAN025
 1103 CONTINUE                                                                  TRAN026
C     TEST FOR NECESSITY OF DUMMY VARIABLES                                     TRAN027
 1181 IF ( ISUMO - ISUMD ) 1115, 1112, 1191                                     TRAN028
C     DUMMY DESTINATION REQUIRED                                                TRAN029
 1191 NDEST = NDEST + 1                                                         TRAN030
      IDEST( NDEST ) = ISUMO - ISUMD                                            TRAN031
```

```
C       ICOST( I,J ) = COST OF SHIPPING A UNIT FROM  I  TO  J               TRAN032
 1101 DO 1104  I = 1, NORIG                                                  TRAN033
      ICOST( I,NDEST ) = 999999                                             TRAN034
 1104 CONTINUE                                                              TRAN035
      GO TO 1112                                                            TRAN036
C     DUMMY ORIGIN REQUIRED                                                 TRAN037
 1115 NORIG = NORIG + 1                                                     TRAN038
      IORIG( NORIG ) = ISUMD - ISUMO                                        TRAN039
 1125 DO 1105  J = 1, NDEST                                                 TRAN040
      ICOST( NORIG,J ) = 999999                                             TRAN041
 1105 CONTINUE                                                              TRAN042
C     IBAS( I,J ) = NUMBER OF UNITS SHIPPED FROM  I  TO  J                  TRAN043
 1112 DO 1117  I = 1, NORIG                                                 TRAN044
      DO 1116  J = 1, NDEST                                                 TRAN045
      IBAS( I,J ) = 0                                                       TRAN046
 1116 CONTINUE                                                              TRAN047
 1117 CONTINUE                                                              TRAN048
 1122 NROW = 1                                                              TRAN049
      NCOL = 1                                                              TRAN050
      ISAV = K1                                                             TRAN051
C     IU( I ) = CODE FOR DEPLETION OF RESOURCES AT ORIGIN  I                TRAN052
C     IV( J ) = CODE FOR SATISFACTION OF REQUIREMENTS AT DESTINATION  J     TRAN053
      DO 1107  I = 1, NORIG                                                 TRAN054
      IU( I ) = 1                                                           TRAN055
 1107 CONTINUE                                                              TRAN056
      DO 1108  J = 1, NDEST                                                 TRAN057
      IV( J ) = 1                                                           TRAN058
 1108 CONTINUE                                                              TRAN059
C     GENERATE INITIAL FEASIBLE SOLUTION - COLUMN MINIMA RULE               TRAN060
 1142 IF ( ICOST( NROW, NCOL ) - ISAV )  1152, 1143, 1143                   TRAN061
 1143 NROW = NROW + 1                                                       TRAN062
 1133 IF ( NROW - NORIG ) 1142, 1142, 1163                                  TRAN063
 1152 IF ( IU( NROW ) - 1 ) 1143, 1162, 1143                                TRAN064
C     ISAV = VALUE OF CURRENT MINIMUM COST                                  TRAN065
```

```
C     NROW1 = ROW NUMBER OF CURRENT MINIMUM COST                              TRAN066
 1162 ISAV = ICOST( NROW, NCOL )                                              TRAN067
      NROW1 = NROW                                                            TRAN068
      GO TO 1143                                                              TRAN069
 1163 IF ( IORIG( NROW1 ) - IDEST( NCOL ) )  1114, 1173, 1164                 TRAN070
C     ORIGIN IS DEPLETED - DESTINATION IS NOT SATISFIED                       TRAN071
 1114 IRAS( NROW1, NCOL ) = IORIG( NROW1 ) + K1                               TRAN0711
 1124 IDEST( NCOL ) = IDEST( NCOL ) - IORIG( NROW1 )                          TRAN072
      IU( NROW1 ) = 0                                                         TRAN073
      NROW = 1                                                                TRAN074
      ISAV = K1                                                               TRAN075
      GO TO 1142                                                              TRAN076
C     ORIGIN IS DEPLETED - DESTINATION IS SATISFIED                           TRAN077
C     ASSIGN AN EPSILON TYPE SHIPMENT PROPERLY                                TRAN078
 1173 IRAS( NROW1, NCOL ) = IORIG( NROW1 ) + K1                               TRAN079
      IU( NROW1 ) = 0                                                         TRAN080
      IV( NCOL ) = 0                                                          TRAN081
      NROW = 1                                                                TRAN082
      ISAV = K1                                                               TRAN083
      IF ( NCOL - NDEST ) 1183, 1211, 1211                                    TRAN084
 1183 IF ( ICOST( NROW, NCOL ) - ISAV ) 1193, 1182, 1182                      TRAN085
 1193 IF ( IU( NROW ) - 1 ) 1182, 1123, 1182                                  TRAN086
 1123 ISAV = ICOST( NROW, NCOL )                                              TRAN087
      NROW1 = NROW                                                            TRAN088
 1182 NROW = NROW + 1                                                         TRAN089
      IF ( NROW - NORIG ) 1183, 1183, 1174                                    TRAN090
C     MAKE AN EPSILON TYPE SHIPMENT                                           TRAN091
 1174 IRAS ( NROW1, NCOL ) = K1                                               TRAN092
      NCOL = NCOL + 1                                                         TRAN093
 1184 IF ( NCOL - NDEST ) 1194, 1194, 1211                                    TRAN094
 1194 NROW = 1                                                                TRAN095
      ISAV = K1                                                               TRAN096
      GO TO 1142                                                              TRAN097
C     ORIGIN IS NOT DEPLETED - DESTINATION IS SATISFIED                       TRAN098
```

```
 1164 IBAS( NROW1, NCOL ) = IDEST( NCOL ) + K1                            TRAN099
      IV( NCOL ) = 0                                                       TRAN100
      IORIG( NROW1 ) = IORIG( NROW1 ) - IDEST( NCOL )                      TRAN101
 1165 NCOL = NCOL + 1                                                      TRAN102
      GO TO 1184                                                           TRAN103
C     DETERMINE SHADOW COSTS FOR CURRENT FEASIBLE SOLUTION                 TRAN104
 1211 DO 1109  I = 1, NORIG                                                TRAN105
C     IU( I ) = VALUE OF  U  FOR ROW I                                     TRAN106
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      IU( I ) = 0                                                          TRAN108
      IU1( I ) = 0                                                         TRAN109
 1109 CONTINUE                                                             TRAN110
      DO 1110  J = 1, NDEST                                                TRAN111
C     IV( J ) = VALUE OF  V  FOR COLUMN J                                  TRAN112
C     IV1( J ) = CODE FOR KNOWLEDGE OF  V  FOR COLUMN J                    TRAN113
      IV( J ) = 0                                                          TRAN114
      IV1( J ) = 0                                                         TRAN115
 1110 CONTINUE                                                             TRAN116
 1221 IU1( 1 ) = 1                                                         TRAN117
      NROW = 1                                                             TRAN118
      NCOL = 1                                                             TRAN119
C     CHECK FOR BASIS ELEMENT                                              TRAN120
 1231 IF ( IBAS( NROW, NCOL ) - K1 )  1232, 1241, 1241                     TRAN121
C     CURRENT ELEMENT IS A BASIS ELEMENT AT 1241                           TRAN122
 1241 IV( NCOL ) = ICOST( NROW, NCOL ) - IU( NROW )                        TRAN123
      IV1( NCOL ) = 1                                                      TRAN124
 1232 NCOL = NCOL + 1                                                      TRAN125
 1242 IF ( NCOL - NDEST ) 1231, 1231, 1252                                 TRAN126
 1252 NROW = NROW + 1                                                      TRAN127
 1262 IF ( NROW - NORIG ) 1272, 1272, 1261                                 TRAN128
 1272 IF ( IU1( NROW ) - 1 ) 1282, 1252, 1282                              TRAN129
 1282 NCOL = 1                                                             TRAN130
 1292 IF ( IBAS( NROW, NCOL ) - K1 ) 1202, 1263, 1263                      TRAN131
 1202 NCOL = NCOL + 1                                                      TRAN132
 1203 IF ( NCOL - NDEST ) 1292, 1292, 1252                           TRAN133
 1263 IF ( IV1( NCOL ) ) 1273, 1202, 1273                            TRAN134
 1273 IU( NROW ) = ICOST( NROW, NCOL ) - IV( NCOL )                  TRAN135
      IU1( NROW ) = 1                                                TRAN136
 1283 NCOL = 1                                                       TRAN137
      GO TO 1231                                                     TRAN138
 1261 NROW = 1                                                       TRAN139
 1271 IF ( IU1( NROW ) ) 1281, 1282, 1281                            TRAN140
 1281 NROW = NROW + 1                                                TRAN141
 1291 IF ( NROW - NORIG ) 1271, 1271, 1213                           TRAN142
C     CALCULATE ALL CELL EVALUATIONS AND FIND LARGEST ONE            TRAN143
 1213 ISAV = 0                                                       TRAN144
      NROW = 1                                                       TRAN145
 1223 NCOL = 1                                                       TRAN146
 1233 IF ( IBAS( NROW, NCOL ) - K1 ) 1234, 1243, 1243                TRAN147
 1243 NCOL = NCOL + 1                                                TRAN148
 1253 IF ( NCOL - NDEST ) 1233, 1233, 1254                           TRAN149
 1254 NROW = NROW + 1                                                TRAN150
 1264 IF ( NROW - NORIG ) 1223, 1223, 1274                           TRAN151
 1274 IF ( ISAV ) 1311, 1480, 1311                                   TRAN152
 1234 IS1 = IU( NROW ) + IV( NCOL ) - ICOST( NROW, NCOL )            TRAN153
 1244 IF ( IS1 - ISAV ) 1243, 1243, 1245                             TRAN154
 1245 ISAV = IS1                                                     TRAN155
      NROW1 = NROW                                                   TRAN156
      NCOL1 = NCOL                                                   TRAN157
      GO TO 1243                                                     TRAN158
C     THE CURRENT SOLUTION IS NOT OPTIMAL                            TRAN159
C     DETERMINE THE NEXT BASIC FEASIBLE SOLUTION                     TRAN160
 1311 K = ( NORIG + NDEST ) * 2                                      TRAN161
C     INET VECTOR STORES THE LABELS FOR THE ELEMENTS IN THE LOOP     TRAN162
      DO 1126 I = 1, K                                               TRAN163
      INET( I ) = 0                                                  TRAN164
 1126 CONTINUE                                                       TRAN165
      DO 1127 I = 1, NORIG                                           TRAN166
      INET1(I) = 0                                                      TRAN167
 1127 CONTINUE                                                          TRAN168
      DO 1128 J = 1, NDEST                                              TRAN169
      INET2( J ) = 0                                                    TRAN170
 1128 CONTINUE                                                          TRAN171
      I = 1                                                             TRAN172
 1321 INET( I ) = NROW1                                                 TRAN173
      INET( I+1 ) = NCOL1                                               TRAN174
      NROW = NROW1                                                      TRAN175
      NCOL = 1                                                          TRAN176
      I = I + 2                                                         TRAN177
 1331 IF ( TBAS( NROW, NCOL ) - K1 ) 1332, 1341, 1341                   TRAN178
C     CURRENT ELEMENT IS A BASIS ELEMENT AT 1341                        TRAN179
 1341 IF ( NCOL - NCOL1 ) 1351, 1332, 1351                              TRAN180
 1351 INET( I ) = NROW                                                  TRAN181
      INET( I+1 ) = NCOL                                                TRAN182
      I = I + 2                                                         TRAN183
      GO TO 1313                                                        TRAN184
C     CURRENT ELEMENT IS NOT A BASIS ELEMENT AT 1332                    TRAN185
 1332 NCOL = NCOL + 1                                                   TRAN186
 1322 IF ( NCOL - NDEST ) 1331, 1331, 1312                              TRAN187
C     ERROR TYPE 1 AT 1312                                              TRAN188
C     NO BASIS ELEMENT IN ROW WHICH CONTAINS ENTERING ELEMENT           TRAN189
 1312 KE = 1                                                            TRAN190
      GO TO 1481                                                        TRAN191
 1313 INET2( NCOL ) = 1                                                 TRAN192
      NROW = 1                                                          TRAN193
 1323 IF ( TBAS( NROW, NCOL ) - K1 ) 1333, 1324, 1324                   TRAN194
 1333 NROW = NROW + 1                                                   TRAN195
 1343 IF ( NROW - NORIG ) 1323, 1323, 1353                              TRAN196
 1353 I = I - 2                                                         TRAN197
 1363 IF ( I ) 1315, 1315, 1373                                         TRAN198
 1373 NROW = INET( I )                                                  TRAN199
      NCOL = INET( I+1 )                                                TRAN200
      INET2( NCOL ) = 0                                                 TRAN201
      GO TO 1375                                                        TRAN202
C     ERROR TYPE 2 AT 1315 - NO BASIS LOOP EXISTS                       TRAN203
 1315 KF = 2                                                            TRAN204
      GO TO 1481                                                        TRAN205
C     CURRENT ELEMENT IS A BASIS ELEMENT AT 1324                        TRAN206
 1324 IF ( NROW - INET( I-2 ) ) 1334, 1333, 1334                        TRAN207
 1334 IF ( INET1( NROW ) ) 1353, 1344, 1353                             TRAN208
 1344 INET( I ) = NROW                                                  TRAN209
      INET( I+1 ) = NCOL                                                TRAN210
      I = I + 2                                                         TRAN211
 1354 IF ( NROW - NROW1 ) 1364, 1355, 1364                              TRAN212
 1355 I = I - 2                                                         TRAN213
      GO TO 1353                                                        TRAN214
 1364 INET1( NROW ) = 1                                                 TRAN215
      NCOL = 1                                                          TRAN216
 1374 IF ( IBAS( NROW, NCOL ) - K1 ) 1375, 1384, 1384                   TRAN217
 1375 NCOL = NCOL + 1                                                   TRAN218
 1365 IF ( NCOL - NDEST ) 1374, 1374, 1371                              TRAN219
 1384 IF ( NCOL - INET( I-1 ) ) 1394, 1375, 1394                        TRAN220
 1394 IF ( INET2( NCOL ) ) 1371, 1395, 1371                             TRAN221
 1395 INET( I ) = NROW                                                  TRAN222
      INET( I+1 ) = NCOL                                                TRAN223
      I = I + 2                                                         TRAN224
 1305 IF ( NCOL - NCOL1 ) 1313, 1411, 1313                              TRAN225
C     NO LOOP FOR ROW NUMBER NROW EXISTS AT 1371                        TRAN226
 1371 I = I - 2                                                         TRAN227
 1381 IF ( I ) 1315, 1315, 1382                                         TRAN228
 1382 NROW = INET( I )                                                  TRAN229
      NCOL = INET( I+1 )                                                TRAN230
      INET1( NROW ) = 0                                                 TRAN231
      GO TO 1333                                                        TRAN232
C     DETERMINE THE VARIABLE WHICH LEAVES THE BASIS                     TRAN233
 1411 I = 3                                                             TRAN234
```

POOR ORIGINAL COPY - BEST AVAILABLE AT TIME FILMED

```
      ISAV = K2                                                         TRAN235
 1421 NROW = INET( I )                                                  TRAN236
      NCOL = INET( I + 1 )                                              TRAN237
 1431 IF ( ISAS( NROW, NCOL ) - ISAV ) 1432, 1441, 1441                 TRAN238
 1432 ISAV = IBAS( NROW, NCOL )                                         TRAN239
      NROW2 = NROW                                                      TRAN240
      NCOL2 = NCOL                                                      TRAN241
 1441 IF ( NCOL - NCOL1 ) 1451, 1442, 1451                              TRAN242
 1451 I = I + 4                                                         TRAN243
 1461 IF ( I - K ) 1421, 1421, 1471                                     TRAN244
C     ERROR TYPE 3                                                      TRAN245
C     BASIS LOOP HAS MORE ELEMENTS THAN THERE ARE IN THE BASIS          TRAN246
 1471 KF = 3                                                            TRAN247
      GO TO 1481                                                        TRAN248
 1442 IF ( ISAV - K2 ) 1443, 1452, 1452                                 TRAN249
C     ERROR TYPE 4                                                      TRAN250
C     NO ELEMENT IN THE BASIS LOOP LESS THAN 10**8                      TRAN251
 1452 KF = 4                                                            TRAN252
      GO TO 1481                                                        TRAN253
 1443 J = -1                                                            TRAN254
      NROW = INET( 1 )                                                  TRAN255
      NCOL = INET( 2 )                                                  TRAN256
      IBAS( NROW, NCOL ) = ISAV                                         TRAN257
      ISAV = ISAV - K1                                                  TRAN258
      I = 3                                                             TRAN259
 1444 NROW = INET( I )                                                  TRAN260
      NCOL = INET( I+1 )                                                TRAN261
 1453 IF ( NROW - NROW2 ) 1463, 1454, 1463                              TRAN262
 1454 IF ( NCOL - NCOL2 ) 1463, 1455, 1463                              TRAN263
 1455 IBAS( NROW, NCOL ) = 0                                            TRAN264
      GO TO 1473                                                        TRAN265
 1463 IBAS( NROW, NCOL ) = IBAS( NROW, NCOL ) + ( J * ISAV )            TRAN266
 1473 J = - J                                                           TRAN267
      I = I + 2                                                         TRAN268
 1493 IF ( NCOL - NCOL1 ) 1444, 1211, 1444                              TRAN269
C      OPTIMUM SOLUTION AT 1480                                         TRAN270
 1480 KF = 5                                                            TRAN280
C      RESTORE AVAILIBILITY AND REQUIREMENTS VECTORS, NORIG, NDEST      TRAN2801
 1481 DO 1484 I = 1, NORIG                                              TRAN281
      DO 1483 J = 1, NDEST                                              TRAN282
      IF ( IBAS(I,J) - K1 ) 1483, 1482, 1482                            TRAN283
 1482 IRAS( I,J) = IBAS( I,J ) - K1                                     TRAN284
 1483 CONTINUE                                                          TRAN285
 1484 CONTINUE                                                          TRAN286
      DO 1487 I = 1, NORIG                                              TRAN287
      IORIG(I) = 0                                                      TRAN288
      DO 1486 J = 1, NDEST                                              TRAN289
      IORIG(I) = IORIG(I) + IRAS(I,J)                                   TRAN290
 1486 CONTINUE                                                          TRAN291
 1487 CONTINUE                                                          TRAN292
      DO 1489 J = 1, NDEST                                              TRAN293
      IDEST(J) = 0                                                      TRAN294
      DO 1488 I = 1, NORIG                                              TRAN295
      IDEST(J) = IDEST(J) + IRAS(I,J)                                   TRAN296
 1488 CONTINUE                                                          TRAN297
 1489 CONTINUE                                                          TRAN298
      NORIG = NORIG1                                                    TRAN299
      NDEST = NDEST1                                                    TRAN300
      RETURN                                                            TRAN301
      END                                                               TRAN302
```