Implementing Vehicle Routing Algorithms

B. L. Golden
T. L. Magnanti
H. Q. Nguyen
Massachusetts Institute of Technology
Cambridge, Massachusetts

ABSTRACT

Heuristic programming algorithms frequently address large problems and require manipulation and operation on massive data sets. The algorithms can be improved by using efficient data structures. With this in mind, we consider heuristic algorithms for vehicle routing, comparing techniques of Clarke and Wright, Gillett and Miller, and Tyagi, and presenting modifications and extensions which permit problems involving hundreds of demand points to be solved in a matter of seconds. In addition, a multi-depot routing algorithm is developed. The results are illustrated with a routing study for an urban newspaper with an evening circulation exceeding 100,000.

I. INTRODUCTION

An essential element of any logistics system is the allocation and routing of vehicles for the purpose of collecting and delivering goods and services on a regular basis. Common examples include newspaper delivery [20], schoolbus routing [5], municipal waste collection [4], fuel oil delivery [16], and truck dispatching in any of a number of industries. The system may involve a single depot or multiple depots; the objectives may be aimed at cost minimization (distribution costs, and vehicle or depot acquisition costs) or service improvement (increasing distribution capacities, reducing distribution time, and related network design issues). Constraints may be imposed upon

(i) the depots (numbers, possible locations, and production capabilities),

(ii) the vehicle fleet (types and numbers of vehicles, and vehicle capacities),

(iii) the delivery points (demand requirements, service constraints on delivery time, and order splitting),

(iv) the routing structure (maximum route time or route distance, link capacities, and preferences for radial routes, or routes with points closer together),

(v) operator scheduling and assignments (union regulations),

and (vi) system dynamics (inventory holdings, and distribution or acquisition lag times).

Applications emphasizing various of these characteristics lead to a continuum of overlapping models including location models such as median and minimax location, scheduling models such as crew scheduling, distribution models such as minimum cost flow or shortest paths, or location-distribution models such as warehouse location. In this paper, we consider models of the routing type with fixed depots, constrained for the most part by fleet, delivery point, and route structure restrictions. Our purposes are three-fold. By synthesizing and extending the results presented in Golden [20] and Nguyen [35], we first formulate integer programming models of these problems, building upon the traveling salesman problem as the "core" model. We then review the most promising heuristic solution procedures for these problems suggested in the literature. Finally, we suggest particularly fast implementations of the Clarke-Wright heuristic procedure. Our method uses efficient data handling capabilities such as heap structures to enhance both computations and storage. Our results indicate that algorithmic modifications of this nature can greatly extend the size of problems amenable to analysis.

We have applied the algorithm to a distribution system for an urban newspaper with an evening circulation exceeding 100,000. This problem contains nearly 600 drop points for newspaper bundles and was solved with 20 seconds of execution time on an IBM 370/168. Fast computations of this nature permit the algorithm to be used as an operational tool and to be used interactively to study modeling assumptions such as demand patterns. It could be used, for example, as in our newspaper study, to investigate performance measures associated with operating policies like the pre-routing of important drop points on special routes. In addition, fast implementation suggests that the heuristic might be used as a subroutine for more general problems in order to incorporate other of the distribution characteristics delineated above, or to perform on-line dynamic routing.

We believe that our computational experience is indicative of implementations for heuristic algorithms in general. Because these algorithms frequently replace computationally expensive or intractable formal optimization procedures by simple, but usually extensive, addition and comparison operations, data organization can be important. Improving data access for each basic operation can lead to order of magnitude improvements in overall running time.

II. FORMULATIONS

The Traveling Salesman Problem

Most vehicle routing models are variants and extensions of the ubiquitous Traveling Salesman Problem; for a very thorough overview of this problem see Bellmore and Nemhauser [3]. Suppose we are given the matrix of pairwise distances or costs d_{ij} between node i and node j for the n nodes 1, 2, ..., n. We assume $d_{ii} = \infty$ for i = 1, 2, ..., n. The problem is to form a tour of the n nodes beginning and ending at the origin, node 1, which gives the minimum total distance or cost. For notation, let

$$\begin{aligned} n &= \text{the number of nodes in the network} \\ V &= \text{the set of nodes} \\ x_{ij} &= \begin{cases} 1 & \text{if arc } (i,j) \text{ is in the tour} \\ 0 & \text{otherwise} \end{cases} \\ d_{ij} &= \text{the distance on arc } (i,j). \end{aligned}$$

An assignment-based formulation of the problem selects the matrix $X = (x_{ij})$ of decision variables solving:

$$\text{Minimize} \quad \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij}\, x_{ij} \tag{1.1}$$

$$\text{subject to} \quad \sum_{i=1}^{n} x_{ij} = b_j = 1 \quad (j = 1, \ldots, n) \tag{1.2}$$

$$\sum_{j=1}^{n} x_{ij} = a_i = 1 \quad (i = 1, \ldots, n) \tag{1.3}$$

$$X = (x_{ij}) \in S \tag{1.4}$$

$$x_{ij} = 0 \text{ or } 1 \qquad (i = 1, \ldots, n;\ j = 1, \ldots, n). \tag{1.5}$$

The set S is selected to prohibit subtour solutions satisfying the assignment constraints (1.2), (1.3), and (1.5). Several alternates have been proposed for S including

(1) $S = \{(x_{ij}): \sum_{i\in Q} \sum_{j\notin Q} x_{ij} \geq 1$ for every nonempty proper subset Q of V$\}$;

(2) $S = \{(x_{ij}): \sum_{i\in Q} \sum_{j\in Q} x_{ij} \leq |Q| - 1$ for every nonempty subset Q of $\{2, 3, \ldots, n\}\}$;

(3) $S = \{(x_{ij}): y_i - y_j + nx_{ij} \leq n - 1$ for $2 \leq i \neq j \leq n$ for some real numbers $y_i\}$.

Note that S contains nearly 2^n subtour breaking constraints in (1) and (2), but only $n^2 - 3n + 2$ constraints in the ingenious formulation (3) proposed by Miller, Tucker, and Zemlin [33]. Algorithmically, the constraints in (2) have proved very useful, however, for Lagrangian approaches to the Traveling Salesman Problem, as initiated by Held and Karp [22], [23]. For other approaches, both heuristic and optimal, see Lin [30], Lin and Kernighan [31], Christofides and Eilon [9], Little [32], and Shapiro [42].

The Traveling Salesman Problem can be interpreted as a vehicle routing model with one depot and with one vehicle whose capacity exceeds total demand. This model can be extended by considering more vehicles, more depots, different vehicle capacities, and additional route restrictions.

The Multiple Traveling Salesman Problem

The Multiple Traveling Salesmen Problem (MTSP) is a generalization of the Traveling Salesman Problem (TSP) and comes closer to accommodating more real world problems; it has proved to be an appropriate model for the problem of bank messenger scheduling, where a crew of messengers picks up deposits at branch banks and returns them to the central office for processing [43].

Given M salesmen and n nodes in a network the MTSP is to find M subtours (each of which includes the origin) such that every node (except the origin) is visited exactly once by exactly one salesman, so that the total distance traveled by all M salesmen is minimal. If we let $a_1 = b_1 = M$ and $a_i = b_j = 1$ for $i \neq 1$ and $j \neq 1$ in model (1.1) - (1.5), we obtain a MTSP formulation.

Three different papers published in 1973 and 1974 independently derived equivalent TSP formulations of the MTSP [2], [37], [43] and consequently showed that the M-salesmen problem is no more difficult than its one-salesman counterpart (see [13] also). The equivalence is obtained by creating M copies of the origin, each connected to the other nodes exactly as was the original origin (with the same distances). The M copies are not connected, or are connected by arcs each with distances exceeding $\sum_{i=1}^{n} \sum_{j=1}^{n} |d_{ij}|$. In this way, an optimal single salesman tour in the expanded network will never use an arc connecting two copies of the origin. Then, by coalescing the copies back into a single node, the single salesman tour decomposes into M subtours as required in the MTSP.

Vehicle Dispatching

The vehicle dispatching problem was first considered by Dantzig and Ramser [12] who developed a heuristic approach using linear programming ideas and aggregation of nodes. The problem is to obtain a set of delivery routes from a central depot to the various demand points, each of which has known requirements, to minimize the total distance covered by the entire fleet. Vehicles have capacities and maximum route time constraints. All vehicles start and finish at the central depot. We will refer to the following formulation for this problem as the generic vehicle routing problem (VRP):

Minimize
$$\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{NV} d_{ij} x_{ij}^{k} \qquad (2.1)$$

subject to
$$\sum_{i=1}^{n} \sum_{k=1}^{NV} x_{ij}^{k} = 1 \qquad (j = 2, \ldots, n) \qquad (2.2)$$

$$\sum_{j=1}^{n} \sum_{k=1}^{NV} x_{ij}^{k} = 1 \qquad (i = 2, \ldots, n) \qquad (2.3)$$

$$\sum_{i=1}^{n} x_{ip}^{k} - \sum_{j=1}^{n} x_{pj}^{k} = 0 \qquad (k = 1, \ldots, NV;\ p = 1, \ldots, n) \qquad (2.4)$$

$$\sum_{i=1}^{n} Q_i \left(\sum_{j=1}^{n} x_{ij}^{k}\right) \leq P_k \qquad (k = 1, \ldots, NV) \qquad (2.5)$$

$$\sum_{i=1}^{n} t_i^{k} \sum_{j=1}^{n} x_{ij}^{k} + \sum_{i=1}^{n} \sum_{j=1}^{n} t_{ij}^{k} x_{ij}^{k} \leq T_k \qquad (k = 1, \ldots, NV) \qquad (2.6)$$

$$\sum_{j=2}^{n} x_{1j}^{k} \leq 1 \qquad (k = 1, \ldots, NV) \qquad (2.7)$$

$$\sum_{i=2}^{n} x_{i1}^{k} \leq 1 \qquad (k = 1, \ldots, NV) \qquad (2.8)$$

$$X \in S \qquad (2.9)$$

$$x_{ij}^{k} = 0 \text{ or } 1 \qquad \text{for all } i, j, k. \qquad (2.10)$$

where

n = number of nodes
NV = number of vehicles
P_k = capacity of vehicle k
T_k = maximum time allowed for a route of vehicle k
Q_i = demand at node i ($Q_1 = 0$)
t_i^k = time required for vehicle k to deliver or collect at node i ($t_1^k = 0$)
t_{ij}^k = travel time for vehicle k from node i to node j ($t_{ii}^k = \infty$)
d_{ij} = shortest distance from node i to node j

$$x_{ij}^{k} = \begin{cases} 1 \text{ if arc } (i,j) \text{ is traversed by vehicle } k \\ 0 \text{ otherwise} \end{cases}$$

X = matrix with components $x_{ij} \equiv \sum_{k=1}^{NV} x_{ij}^{k}$, specifying connections regardless of vehicle type.

Equation (2.1) states that total distance is to be minimized. Alternatively, we could minimize costs by replacing d_{ij} by a cost coefficient c_{ij}^{k} which depends upon the vehicle type. Equations (2.2) and (2.3) ensure that each demand node is served by one vehicle and only one vehicle. Route continuity is represented by equations (2.4), i.e., if a vehicle enters a demand node, it must exit from that node. Equations (2.5) are the vehicle capacity constraints; similarly, equations (2.6) are the total elapsed route time constraints. For instance, a newspaper delivery truck may be restricted from spending more than one hour on a tour in order that the maximum time interval from press to street be made as short as possible. Equations (2.7) and (2.8) make certain that vehicle availability is not exceeded. Finally, the subtour-breaking constraints (2.9) can be any of the equations specified previously. Since (2.2) and (2.4) imply (2.3), and (2.4) and (2.7) imply (2.8), from now on we consider the generic model to include (2.1) - (2.10) excluding (2.3) and (2.8), which are redundant. We assume that $\max\limits_{1 \leq i \leq n} Q_i < \min\limits_{1 \leq k \leq NV} P_k$. That is, the demand at each node does not exceed the capacity of any truck. Observe that when vehicle capacity constraints (2.5) and route time constraints (2.6) are nonbinding, and can be ignored, this model reduces to a multiple traveling salesman problem by eliminating constraints (2.4) and substituting $x_{ij} = \sum\limits_{k=1}^{NV} x_{ij}^{k}$ in the objective function and remaining constraints.

In our generic model we have assumed that when a demand node is serviced, its requirements are satisfied. A more cumbersome mixed integer programming heterogeneous fleet problem formulation was given in 1957 by Garvin [16] in which this assumption is relaxed. Balinski and Quandt [1] provide another formulation in terms of a set covering problem.

Observe that since $\sum\limits_{j=2}^{n} x_{1j}^{k}$ is 1 or 0 depending upon whether or not the kth vehicle is used, a fixed acquisition cost $f_k \sum\limits_{j=2}^{n} x_{1j}^{k}$ can be added to the model when it is formulated with an objective function to investigate trade-offs between routing and acquisition costs.

We can, without difficulty, generalize the generic model to the multicommodity case where several different types of products must be routed simultaneously over a network in order to satisfy demands at delivery points for the various products [20].

Furthermore, we can incorporate timing restrictions into the vehicle dispatching model. If we define a_j as the arrival time at node j then restrictions on delivery deadlines $\bar{a}_j$ and earliest delivery times $\underline{a}_j$ can be represented by the following nonlinear equations:

$$a_j = \sum_k \sum_i (a_i + t_i^k + t_{ij}^k)\, x_{ij}^k \qquad (j = 1, \ldots, n)$$

$$a_1 = 0$$

$$\underline{a}_j \le a_j \le \bar{a}_j \qquad (j = 2, \ldots, n).$$

Alternatively, the above nonlinear constraints can be replaced by the linear constraints

$$\left.\begin{aligned} a_j &\ge (a_i + t_i^k + t_{ij}^k) - (1 - x_{ij}^k)\, T \\ a_j &\le (a_i + t_i^k + t_{ij}^k) + (1 - x_{ij}^k)\, T \end{aligned}\right\} \text{ for all } i, j, k,$$

where $T = \max_{1 \le k \le NV} T_k$. When $x_{ij}^k = 0$, these constraints are redundant. When $x_{ij}^k = 1$, they determine a_j in terms of the arrival time a_i at the node i preceding node j on a tour, the delivery time t_i^k at node i, and the travel time t_{ij}^k between nodes i and j. Of course, these constraints add considerably to the size of the model (2.1) - (2.10).

Multi-Depot Vehicle Routing

The integer programming formulation of the vehicle routing problem is altered in minor ways to incorporate multiple depots.

Letting nodes 1, 2, ..., M denote the depots, we obtain the formulation by changing the index in constraints (2.2) and (2.3) to (j=M+1,...,n), by changing constraints (2.7) and (2.8) to

$$\sum_{i=1}^{M} \sum_{j=M+1}^{n} x_{ij}^{k} \leq 1 \qquad (k = 1, \ldots, NV) \qquad (2.7')$$

$$\sum_{p=1}^{M} \sum_{i=M+1}^{n} x_{ip}^{k} \leq 1 \qquad (k = 1, \ldots, NV) \qquad (2.8')$$

and by redefining our previous choices for the subtour breaking set S to be:

(1') $S = \{(x_{ij}): \sum_{i\varepsilon Q} \sum_{j \not\varepsilon Q} x_{ij} \geq 1$ for every proper subset Q of V containing nodes 1, 2, ..., M$\}$;

(2') $S = \{(x_{ij}): \sum_{i\varepsilon Q} \sum_{j\varepsilon Q} x_{ij} \leq |Q| - 1$ for every nonempty subset Q of $\{M+1, M+2, \ldots, n\}\}$;

(3') $S = \{(x_{ij}): y_i - y_j + nx_{ij} \leq n - 1$ for $M+1 \leq i \neq j \leq n$ for some real numbers $y_i\}$.

Concluding Comments on Formulations

Hopefully, the formulations discussed in this section provide a unified basis for viewing vehicle routing problems as generalized TSP's. The formulations indicate, for example, that Lagrangian approaches similar to those used by Held and Karp [22], [23] can be applied to these problems by dualizing with respect to constraints (2.2) - (2.8). Procedures of this nature might prove useful when combined with good heuristics. In particular, since the heuristic procedures provide feasible solutions to these problems and the Lagrangian approaches provide bounds to the minimum distance solution, the heuristic and dual approaches together bracket the optimal value for the objective function. This bracketing might conceivably help to evaluate the effectiveness of the heuristic approaches, or might be useful in obtaining optimal solutions to the problems via branch and bound methods.

III. HEURISTIC SOLUTION TECHNIQUES FOR SINGLE DEPOT PROBLEMS

Proposed techniques for solving vehicle routing problems have fallen into two classes - those which solve the problem optimally by branch and bound techniques, and those which solve the problem heuristically. Since the optimal algorithms have been viable only for very small problems, we concentrate on heuristic algorithms. Christofides claims that the largest vehicle routing problem of any complexity that has been solved exactly involved only 23 customers [10]. The three vehicle routing heuristic methods which we will mention (Clarke and Wright [11], Tyagi [46], and Gillett and Miller [19]) have been used for problems with up to 1000 customers.

The Clarke-Wright algorithm is an "exchange" algorithm in the sense that at each step one set of tours is exchanged for a better set of tours. Initially, we suppose that every two demand points i and j are supplied individually from two vehicles (refer to Figure 1 below).

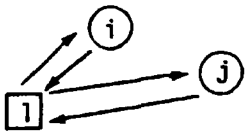

Fig. 1 Initial setup.

Now if instead of two vehicles, we used only one, then we would experience a savings in travel distance of $(2d_{1i} + 2d_{1j}) - (d_{1i} + d_{1j} + d_{ij}) = d_{1i} + d_{1j} - d_{ij}$ (see Figure 2 below).

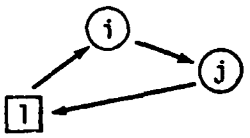

Fig. 2 Nodes i and j have been linked.

For every possible pair of demand points i and j there is a corresponding savings s_{ij}. We order these savings from greatest to least and starting from the top of the list we link nodes i and j where s_{ij} represents the current maximum savings unless the problem constraints are violated. Christofides and Eilon found from 10 small test problems that tours produced from the "savings" method averaged only 3.2 percent longer than the optimal tours [8].

Tyagi [46] presents a method which groups demand points into tours based on a "nearest neighbor" concept. That is, points are added to a tour sequentially, each new addition being the closest point to the last point added to the tour. Having grouped the delivery points into m tours, the vehicle dispatching problem reduces to m Traveling Salesman Problems, one for each tour. Further details are provided in Tyagi [46] and Golden, Magnanti, and Nguyen [2I].

In a recent paper, Gillett and Miller [19] introduce an efficient buildup algorithm for handling up to about 250 nodes. Rectangular coordinates for each demand point are required, from which we may calculate polar coordinates. We select a "seed" node randomly. With the central depot as the pivot, we start sweeping (clockwise or counterclockwise) the ray from the central depot to the seed. Demand nodes are added to a route as they are swept. If the polar coordinate indicating angle is ordered for the demand points from smallest to largest (with seed's angle 0) we enlarge routes as we increase the angle until capacity restricts us from enlarging a route by including an additional demand node. This demand point becomes the seed for the following route. Once we have the routes, we can apply TSP algorithms such as the Lin-Kernighan heuristic to improve tours and obtain significantly better results. In addition, we can vary the seed and select the best solution.

Tyagi's method and Gillett and Miller's method are essentially "cluster first - route second" (see Bodin [6]) approaches. The Clarke and Wright algorithm can be viewed similarly, especially when the resulting tours are improved via the Lin-Kernighan TSP heuristic; Robbins, Shamblin, Turner and Byrd [40] provide computational experience for such a hybrid approach. Newton and Thomas [34] have proposed a "route first - cluster second" approach which we do not explore in this paper.

The Clarke and Wright algorithm overcomes a major deficiency of the other two algorithms in that demand points farther away from the central depot are considered early for linking. This property means that few of these "problem" nodes are left to be grouped at the end when there are virtually no degrees of freedom.

The literature contains computational experience relating to the Clarke and Wright [11], and the Gillett and Miller [19] algorithms. On a 50-customer problem (which will be mentioned later) the former algorithm had computation time of 6 seconds on an IBM 7090, the latter, 120 seconds on an IBM 360/67. However, the second objective value was slightly lower. A 250-location problem with an average of 10 locations per route was solved in just under 10 minutes of IBM 360/67 time using the Gillett and Miller algorithm. The Tyagi algorithm has been programmed by

Klincewicz [28] on an IBM 370/168. The same 50-node problem was solved in 1 second of execution time yielding a rather high objective value. Several experiments with the Tyagi algorithm have confirmed the poor objective performance of this approach. Yellow's modification of the Clarke-Wright procedure appears computationally to be quite powerful; problems of 200 nodes have been solved in less than a minute and a problem with 1000 nodes was solved in five minutes on an IBM 360/50 [50]. Webb, in applying a similar sequential savings approach, reports having solved 400-customer problems in less than 6 seconds on a CDC 6600 [47]. Several basic questions arise since implementation is not discussed in the paper at all:

(i) What operations are included in this 6-second figure?
(ii) What are the details of data storage and manipulation in this implementation?
(iii) What kinds of vehicle routing problems can be handled?

IV. A MODIFIED CLARKE-WRIGHT ALGORITHM WHICH IS VERY FAST

Probably the most popular of the heuristic solution techniques discussed in the last section is the Clarke-Wright "savings" method which IBM has programmed as VSPX [25], a flexible computer code to handle complex routing problems.

At each step in the Clarke-Wright algorithm we seek the greatest positive savings s_{ij} subject to the following restrictions:

(i) nodes i and j are not already on the same route;
(ii) neither i nor j are interior to an existing tour;
(iii) vehicle availability is not exceeded;
(iv) vehicle capacity is not exceeded;
(v) maximum number of drops (or maximum route time) is not exceeded.

In many applications, where the delivery or pickup time is a sizeable portion of total route time, it makes sense to limit the number of drops rather than the route time since travel times are so difficult to estimate. Nodes i and j, then, are linked together to form a new route, and the procedure is repeated until no further savings are possible.

There have been modifications made to the basic approach, most notably those suggested by Beltrami and Bodin [4] and by Yellow [50]. In this paper, we emphasize data structures and list processing and discuss a new implementation of the Clarke-Wright algorithm which is motivated by (i) optimality considerations, (ii) storage considerations, and (iii) sorting considerations and program running time. We modify the basic Clarke-Wright algorithm in three ways:

(1) by using a route shape parameter γ to define a modified savings

$$s_{ij} = d_{1i} + d_{1j} - \gamma d_{ij}$$

and finding the best route structure obtained as the parameter is varied;

(2) by considering savings only between nodes that are "close" to each other;

(3) by storing savings s_{ij} in a heap structure to reduce comparison operations and ease access.

Route Shape Parameter

The route shape parameter was introduced by Yellow [50], and is an outgrowth of an algorithm developed by Gaskell [17]. When the parameter γ increases from zero, greater emphasis is placed on the distance between points i and j rather than their position relative to the central depot. The search for the best route structure as γ is varied provides heuristic solutions which are closer to the optimal solution than otherwise obtained via the traditional algorithm where $\gamma = 1$. For example, in applying the modified algorithm to the 50-node problem in Christofides and Eilon [8], a solution of total distance 577 was obtained with a route shape parameter of 1.3, whereas the traditional Clarke-Wright solution is 585. Perhaps more importantly, the search over γ promotes flexibility. In most applications there are "unstated" goals and/or constraints. By varying the route shape parameter we can produce several "sufficiently good" tentative solutions from which a final solution may be chosen on the basis of these unstated considerations.

Storage

The Clarke-Wright algorithm was designed initially to handle a matrix of real inputs, distances and savings. One might argue that these variables can be rounded to integers to simplify computations. However, we felt that precision was important. In dealing with a 600-node problem a matrix form would require 360,000 storage locations for these inputs. This assumes an undirected network with symmetric distances in order that the d_{ij} terms can fill above the diagonal in the matrix, and the s_{ij} terms can fill below. Of course, if we use only the traditional Clarke-Wright algorithm we need not store the distances at all. At least some distances must be kept in memory, however, in order to vary the route shape parameter (in particular, the distances from the origin to all other nodes must be stored).

Rather than consider all pairwise linkings, as in a matrix approach, we can become more selective and work with the linkings of greatest interest and convenience. We have been investigating a newspaper distribution problem of nearly 600 nodes; our computer system cannot set aside 360,000 internal storage locations (more than 1.5 million bytes). And if it were possible, it would be inefficient to calculate savings which could never be realized. Most of the potential linkings are infeasible from a practical standpoint and our modified Clarke-Wright program reflects this observation. The topology of the network is stored in the following manner. In general, for each arc we record its origin node, its destination node, and its length. The approach requires 3A rather than n^2 storage locations where n is the total number of nodes and A the number of arcs under consideration. For undirected networks we can order the arcs lexicographically and save storage.

Given the x and y coordinates of the nodes in the transportation network, we can superimpose an artificial grid of width WIDTH and height HEIGHT over the network. Furthermore, the grid can be divided arbitrarily into DIV^2 rectangles in such a way that each demand node is contained in a rectangle of width WIDTH/DIV and height HEIGHT/DIV. Node I has box coordinates BX(I) and BY(I). The set of arcs we will be working with includes all arcs between the central depot (node 1) and other nodes (demand nodes), and all arcs between nodes which are no further apart than one box. In other words, if $|\text{BX(I)} - \text{BX(J)}| > 1$ or $|\text{BY(I)} - \text{BY(J)}| > 1$ the arc (I,J) is ignored. The value of DIV influences the accuracy of the heuristic algorithm and should be altered according to the problem. If the number of demand points, $n-1$, is small DIV should be small; if n is large DIV should be large. The smaller the parameter DIV, for a particular problem, the larger is the number of arcs to be considered.

Heap Structure

At each step of the algorithm, we must determine the maximum savings. This comparison of savings can be accomplished quickly and conveniently by partially ordering the data in a heap structure and updating the structure at each step after altering the routes. More precisely, the savings s_{ij} as denoted by $s_1, s_2, \ldots, s_m$ are arranged in a binary tree with k levels called a heap. The essential property of a heap is that $s_i \geq s_{2i}$ and $s_i \geq s_{2i+1}$. Below is a heap for k = 3 ($s_1 = 25$, $s_2 = 21$, $s_3 = 16$, and so on).

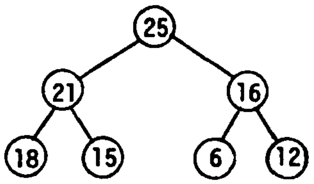

Fig. 3

If the list does not completely fill the last level of the tree, then we can add positions in the last level with entries of $-\infty$. Clearly, s_1 corresponds to the maximum savings possible of those savings under consideration.

First, the subroutine STHEAP of our implementation arranges $s_1, s_2, \ldots, s_m$ into a heap using the ideas in Williams [48] and Floyd [14]. STHEAP builds a heap in $O(m\log_2 m)$ comparisons and interchanges in the worst case. Now, suppose that s_1 corresponds to arc (i,j) and that neither i nor j become interior points on a route upon linking nodes i and j. Setting s_{ij} to zero we, in effect, remove s_1 from the heap since only positive savings are considered. Actually, we bubble s_1 down the heap until it finds its new home. A new heap can be constructed in this case with remarkable ease by the subroutine OTHEAP, since s_1 is always made smaller and must therefore move down the binary tree. If s_1 has already been moved to position k then the new entry s_k is compared with its sons only (s_{2k} and s_{2k+1}) until $s_k \geq \max\{s_{2k}, s_{2k+1}\}$

or the entry s_k is at the lowest level of the heap. If $s_k \leq s_m = \max\{s_{2k}, s_{2k+1}\}$ then s_k and s_m are interchanged, and s_k is compared again with its two new sons.

If node i (or j) becomes an interior point on a route when nodes i and j are linked, then we no longer try linking node i (or j) with any other nodes in the network. Conceptually, we can cross all entries s_r off the heap, where s_r is the savings incurred by linking an interior node to another node. We accomplish this task by (i) setting each such entry to zero and (ii) having subroutine OTHEAP reconstruct a heap structure starting from position r. OTHEAP is called as many times as there are adjacent nodes to the interior node i (or j).

The first entry of the heap always represents the most promising link to add to the current routes, and by eliminating infeasible linkings quickly by OTHEAP, the algorithm proceeds very rapidly until no additional savings can be realized. OTHEAP, exploits the simple observation that when an entry is changed it is always decreased to zero.

Implementing the STHEAP and OTHEAP procedures effectively requires data structures which point from the heap structure to the location of the corresponding arc data in the original problem data. These pointers are being updated throughout the computer program as the heap structure is altered. Golden, Magnanti, and Nguyen provide further details [21].

Computational Results

The computer program has been programmed in the WATFIV version of FORTRAN. Computational studies have been especially encouraging. A 50-node problem (mentioned earlier) was solved on the IBM 370/168 at M.I.T. using less than one second total execution time, including all input and output operations. Nodes were read in by coordinates, so all distances were calculated within the program. Realistic data for a newspaper distribution problem was gathered for us by a local newspaper. The evening edition has a city circulation of about 100,000 with nearly 600 demand points where bundles of newspapers are delivered. The total execution time was 20 seconds and the routes produced were considered from reasonable to very interesting by the circulation department involved. We believe our program can be of considerable value as a tool in helping to alleviate newspaper distribution problems. It is of interest that a major portion of the 20 seconds involved the determination of node adjacencies and distances.

Below we report sensitivity analyses for the newspaper problem with respect to the following parameters:

(i) box sizes (Table I);
(ii) vehicle capacities (Table II);
(iii) maximum number of drops on a tour (Table III);
(iv) route shape parameter (Table IV).

In each case, only one parameter is varied at a time. The tendencies exhibited in Tables I, II, and III agree with our intuition, whereas Table IV is a bit more interesting. The best choice for route shape parameter appears to be truly problem dependent (in this example, $\gamma = 1.0$ and $\gamma = .4$ are strong candidates, taking both objective value and number of tours into account). Several parameters might be varied simultaneously in future parametric studies. This parametric analysis affords us the opportunity to choose the route structure most appealing in light of various (often conflicting) objectives.

Value of DIV	Total Distance Traveled	Number of Routes
30	306.19	22
25	272.55	17
24	259.67	15
23	267.11	17
22	261.83	16

Table I Testing box sizes.

Vehicle Capacity	Total Distance Traveled	Number of Routes
3000	266.20	18
3500	262.32	17
4000	262.32	17
4500	262.32	17
5000	262.32	17

Table II Testing vehicle capacities (the capacity of the smaller of two types of vehicles was varied).

Maximum Number of Drops	Total Distance Traveled	Number of Routes
60	262.32	17
70	262.65	16
80	260.36	15
90	259.86	15
100	259.67	15

Table III Testing the maximum number of drops on a tour.

Value of γ	Total Distance Traveled	Number of Routes
.4	263.09	15
.6	264.72	16
.8	266.03	17
1.0	262.32	17
1.2	277.46	18
1.4	279.77	19
1.6	276.99	20

Table IV Testing the route shape parameter.

V. HEURISTIC SOLUTION TECHNIQUES FOR MULTI-DEPOT PROBLEMS

Whereas the VRP has been studied widely, the multi-depot problem has attracted less attention. The relevant literature is represented by only a few papers. The exact methods for the single depot VRP can be extended to the general case, but as in the case for the single depot VRP, because of storage and computation time, only small problems can be dealt with currently. Problems arising in practice are beyond the scope of these optimal algorithms. Three relatively successful heuristic approaches have been developed for the multi-depot problem.

A first class of heuristics generates one solution arbitrarily and proceeds to improve the solution by exchanging nodes one at a time between routes until no further improvement is possible. Some typical examples are Wren and Holliday [49] and Cassidy and Bennett [7].

The Wren and Holliday program has been run on Gaskell's ten sample cases and the authors report results comparable with other methods with respect to computation time and accuracy. No tests involving more than 100 nodes are reported. Cassidy and Bennett report a successful case study using a similar approach.

The two other approaches are extensions of heuristics developed for the single depot VRP. The Gillett and Johnson algorithm [18] is an extension of the Gillett and Miller "sweep" algorithm discussed previously. The method solves the multi-depot problem in two stages. First, locations are assigned to depots. Then, several single depot VRP's are solved. Each stage is treated separately.

Initially, all locations are in an unassigned state. For any node i, let

t'(i) be the closest depot to i, and

t"(i) be the second closest depot to i.

for each node i, the ratio

$$r(i) = d_{i,\, t'(i)} / d_{i,\, t''(i)} \tag{3}$$

is computed and all nodes are ranked by increasing values of r(i). The arrangement determines the order in which the nodes are assigned to a depot: those which are relatively close to a depot are considered first. After a certain number of nodes have been assigned from the list of r(i), a small cluster is formed around every depot.

Locations i such that the ratio r(i) is close to 1, are assigned more carefully. If two nodes j and k are already assigned to a depot t, inserting i between j and k on a route linked to t creates an additional distance equal to

$$d_{ji} + d_{ik} - d_{jk}$$

which represents a part of the total distance (or cost). In other words, the algorithm assigns location i to a depot t by inserting i between two nodes already assigned to depot t, in the least costly manner.

The Sweep Algorithm is used to construct and sequence routes in the cluster about each depot independently. A number of refinements are made to improve the current solution.

Tillman and Cain's [44] modified version of the Clarke-Wright algorithm starts with the initial solution consisting of servicing each node exclusively by one route from the closest depot. Let

d_{ij} = distance between demand nodes i and j, and

d_i^k = distance between node i and depot k.

The total distance of all routes is then $D = \sum_{i=1}^{N} 2 \min_k \{d_i^k\}$.
The method successively links pairs of nodes in order to decrease the total distance traveled. One basic rule is assumed in the algorithm: the first assignment of nodes to the nearest terminal is temporary but once two or more nodes have been assigned to a common route from a terminal, the nodes are not reassigned to another terminal. In addition, as in the original Clarke-Wright algorithm, nodes i and j can be linked only if neither i nor j are interior to an existing tour.

At each step, the choice of linking a pair of nodes i and j on a route from terminal k is made in terms of two criteria:

(i) a savings when linking i and j at k,
(ii) a penalty for not doing so.

Nodes i and j can be linked only if no constraints are violated.

The computation of savings is not as straightforward as in the case of a single depot. The savings s_{ij}^k are associated to every combination of demand nodes i and j and depot k, and represent the decrease in total distance traveled obtained when linking i and j at k. The formula is given by

$$s_{ij}^k = \tilde{d}_i^k + \tilde{d}_j^k - d_{ij} \tag{4}$$

where $$\tilde{d}_i^k = \begin{cases} 2 \min_t \{d_i^t\} - d_i^k & \text{if i has not yet been given a permanent assignment} \\ d_i^k & \text{otherwise.} \end{cases}$$

In the first case, node i is being reassigned from its closest depot and the previously assigned distance $2 \min_t \{d_i^t\}$ is saved;

in the second case, node j is being inserted between depot k and node i and the link from i to k is dropped from the current solution. Nguyen [35] illustrates formula (4) with an example.

The savings s_{ij}^{k} are computed for i, j = 1, ..., N $(i \neq j)$ and k = 1, ..., M at each step. They can be stored in M matrices, each N by N.

Tillman and Cain [44] introduce a penalty correction factor to the Clarke-Wright procedure as follows. Suppose that nodes i and j are linked on a route from depot k at a particular iteration with savings s_{ij}^{k}. If, instead, nodes i and u were linked on a route from depot m at a later iteration, the savings would be s_{iu}^{m}. Assuming that s_{ij}^{k} is selected to give maximum savings out of node i, $s_{iu}^{m} \leq s_{ij}^{k}$ and $O_{iu}^{m} \equiv s_{ij}^{k} - s_{iu}^{m}$ measures the opportunity cost of using this later link addition. Similarly, if at a later iteration, node j were linked to node v on a route from depot m, then $O_{vj}^{m} \equiv s_{ij}^{k} - s_{vj}^{m}$ would be the corresponding opportunity cost. Using these observations, Tillman and Cain define a penalty p_{ij}^{k} for not selecting link (i,j) on a route from depot k at the current iteration by:

$$p_{ij}^{k} = \min \{O_{iu}^{m} \mid \text{all } (u,m),\ u=1,\ldots,N;\ m=1,\ldots,M, \text{ except } (j,k)\}$$
$$+ \min \{O_{vj}^{m} \mid \text{all } (v,m),\ v=1,\ldots,N;\ m=1,\ldots,M, \text{ except } (i,k)\}.$$

In order to incorporate the concepts of savings and penalties in a simple way, Tillman and Cain suggest the expression

$$f_{ij}^{k} = \alpha\, s_{ij}^{k} + \beta\, p_{ij}^{k}$$

where α and β are two selected (or varied) positive weights. At each step, the link (i,j) at k is chosen which maximizes f_{ij}^{k} and which yields a feasible solution (with regard to capacity and maximum route time restrictions).

A penalty function can obviously be applied to the single depot VRP as well, to improve upon Clarke and Wright's original algorithm. The latter has the drawback of proceeding very myopically and the consideration of a penalty function might help much in overcoming this drawback. The penalty calculations, however, impose additional computational burdens.

VI. ALGORITHM I FOR THE MULTI-DEPOT VRP

The algorithm proposed here is based on the savings method, and is motivated by a desire to find an algorithm capable of handling large size problems. Algorithm I uses Tillman and Cain's approach for computing savings but excludes the idea of a penalty function. Our main contribution, at this stage, is a method of manipulating data which allows fast computation and minimizes storage requirements.

In most codes written for the VRP using the savings approach, the necessity of storing the list of savings often has set the limit on the size of problems that is tractable to solve. In the Tillman and Cain approach to the multi-depot VRP, to each depot corresponds a matrix of savings and the total number of required storage locations grows quickly (even though the number of depots is generally small for transportation networks). In order to cope with the difficulty of accessing and storing such massive data sets efficiently, we have

(1) limited the amount of data that must be manipulated at each step, and
(2) exploited the symmetry of savings data.

The first objective has been met in two ways. First we have superimposed a grid structure, as described previously for the single depot VRP, whereby we only consider joining nodes in adjacent boxes of the grid. In order to assess the economy of computation brought about by the grid, we have tried the idea on a problem with 50 nodes and 4 depots: without using a grid, Algorithm I took 2.04 seconds total CPU time and with a grid, it took 1.38 seconds while giving the same solution on an IBM 370/168.

Second, we have not manipulated all of the savings s_{ij}^k at each step of the algorithm. Instead, we have defined ROWMAX (I) as the largest element in row I of all of the M matrices $S^k = (s_{ij}^k)$ for $k = 1, 2, \ldots, M$. At each step of the algorithm, the maximal savings has been obtained by searching for the maximum among the ROWMAX (I) for $I = 1, \ldots, N-1$. This can be accomplished readily via heap structures, as we have noted previously. The main advantage of using the auxiliary variable

ROWMAX and searching for the maximal savings among ROWMAX (I) for I = 1, ..., N-1 instead of searching directly through the whole list of savings is an economy of computation time. At each iteration of the algorithm, we have to update some rows and columns of the savings matrix. Typically, this operation involves changes only for a few elements of the vector ROWMAX. Updating the vector ROWMAX and then selecting its largest element involves considerably fewer comparisons than searching through the whole list of new savings.

To conserve storage, we can utilize the fact that the savings s_{ij}^{k} associated with the linkage of nodes i and j at depot k are symmetric in i and j. We can exploit this symmetry by storing savings in rectangular rather than square matrix form and thereby eliminate redundancy. This is accomplished by splitting the savings matrix (s_{ij}^{k}) in half by a vertical cut and "folding" the subdiagonal elements to the right of the cut to fit in the superdiagonal elements to the left of the cut. Figures 4 and 5 illustrate this storage scheme. Note that the diagonal elements of (s_{ij}^{k}) are always zero and do not have to be stored. Also, note that the storage scheme is slightly different when N is even and N is odd.

Each element S^{k} (I,J) of the matrix S^{k} that we are storing corresponds to a savings s_{ij}^{k} which is stored in location (I,J) of S^{k} according to the following rules:

$$\text{For N even,} \quad S^{k}(I,J) = \begin{cases} s^{k}_{I+1,\ J} & \text{if } I \geq J \\ s^{k}_{N-I+1,\ N-J+1} & \text{if } I < J; \end{cases}$$

$$\text{For N odd,} \quad S^{k}(I,J) = \begin{cases} s^{k}_{I+1,\ J} & \text{if } I \geq J \\ s^{k}_{N-I+1,\ N-J+1} & \text{if } I < J \leq (N-1)/2 \\ 0 & \text{if } I < J = (N+1)/2. \end{cases}$$

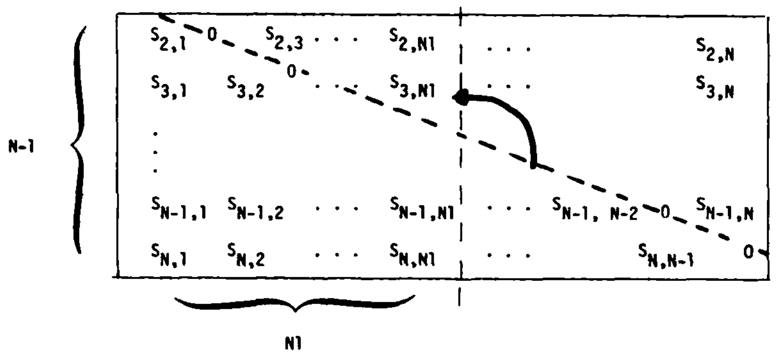
N-1
N1
$s_{2,1}$ 0 $s_{2,3}$ . . . $s_{2,N1}$ . . . $s_{2,N}$
$s_{3,1}$ $s_{3,2}$ 0 . . . $s_{3,N1}$ . . . $s_{3,N}$
$s_{N-1,1}$ $s_{N-1,2}$ . . . $s_{N-1,N1}$ . . . $s_{N-1,N-2}$ 0 $s_{N-1,N}$
$s_{N,1}$ $s_{N,2}$ . . . $s_{N,N1}$ . . . $s_{N,N-1}$ 0

Fig. 4 Storing savings in rectangular form (note N1 = N/2 if N is even and N1 = (N+1)/2 if N is odd).

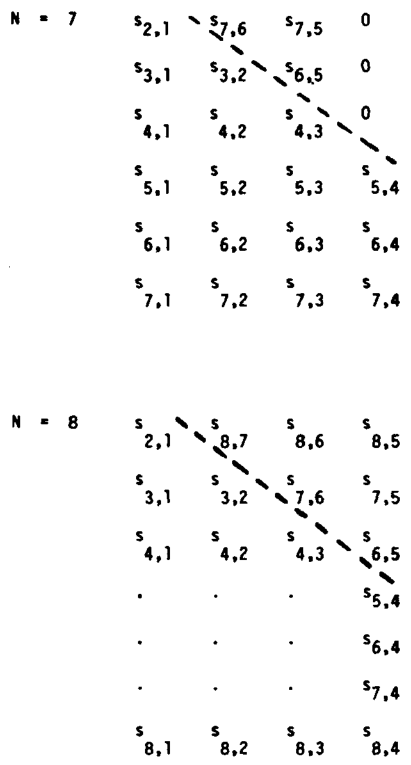
N = 7
$s_{2,1}$ $s_{7,6}$ $s_{7,5}$ 0
$s_{3,1}$ $s_{3,2}$ $s_{6,5}$ 0
$s_{4,1}$ $s_{4,2}$ $s_{4,3}$ 0
$s_{5,1}$ $s_{5,2}$ $s_{5,3}$ $s_{5,4}$
$s_{6,1}$ $s_{6,2}$ $s_{6,3}$ $s_{6,4}$
$s_{7,1}$ $s_{7,2}$ $s_{7,3}$ $s_{7,4}$
N = 8
$s_{2,1}$ $s_{8,7}$ $s_{8,6}$ $s_{8,5}$
$s_{3,1}$ $s_{3,2}$ $s_{7,6}$ $s_{7,5}$
$s_{4,1}$ $s_{4,2}$ $s_{4,3}$ $s_{6,5}$
$s_{5,4}$
$s_{6,4}$
$s_{7,4}$
$s_{8,1}$ $s_{8,2}$ $s_{8,3}$ $s_{8,4}$

Fig. 5 Examples of savings matrices.

A competing storage scheme involves storing the savings associated with two depots together. The upper half of the k^{th} matrix contains entries associated with savings for depot 2k - 1, and the lower half of the matrix contains savings associated with depot 2k as shown in Figure 6.

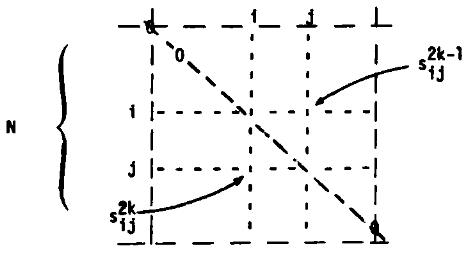

Fig. 6 Storing the savings for two depots together.

This scheme is not as compact as the previous method of storage, since zero savings associated with s_{ii}^{2k} or s_{ii}^{2k-1} are kept along the diagonal. Moreover, the last matrix in the storage scheme will be only half filled with savings data when the number of depots is odd. Nevertheless, in testing both storage schemes, we have found that this second method leads to a slightly faster implementation, since less manipulation of indices is required.

For large problems, a further economy is achieved by transforming the s_{ij}^{k} into half-word integers, caution being taken against overflow. Round-off errors can be minimized by first multiplying all real distances by a factor of 100.

Some Computational Results

The code we have written uses the second storage scheme described above and has been compiled in 0.18 seconds on an IBM 370/168; it requires 8400 bytes of core. A problem of 200 nodes and 4 depots necessitates 200k of core memory (using half-word integers for savings), 300 nodes and 4 depots take 400k, 200 nodes and 10 depots take 420k. We consider these to be medium-sized problems. Results for some sample problems are summarized in Table VI.

As with the single depot VRP a route shape parameter line search has been studied. In general there seems to be no a priori method for determining a best value of γ for any given problem. The algorithm is fast enough, however, so that we can try different values of γ and then select the best solution produced. We have tried the idea on a 50-node, 4-depot problem; results are given in Table V.

Value of γ	Solution Produced by Algorithm	
	Total Distance Traveled	Number of Routes
.2	584.63	6
.4	559.89	6
.6	512.09	6
.8	509.77	6
1.0	509.30	7
1.2	508.07	7
1.4	518.10	7
1.6	536.40	11
1.8	560.40	13
2.0	587.28	15

Table V Testing route shape parameter for multi-depot problems.

The advantage of this approach, as we have pointed out previously, is that we are given a few alternatives to choose from and depending on the objective function, one solution or another can be selected. For example, in the above illustration, the sole criterion of minimal distance traveled would determine the choice of $\gamma = 1.2$; but the consideration of distance traveled combined with the number of routes might set the value $\gamma = .8$ as the best choice.

Another positive feature of the method is that after drawing the different routes produced for different values of γ, it is usually possible to combine some of them manually, simply by examining the various alternatives, to produce an even better solution. For example, we worked a few minutes on the solution obtained with $\gamma = 1.0$, and obtained a solution with 6 routes and a total distance traveled of 487 units. We will describe a computer code which performs a similar refining operation later in this paper.

The above idea is of course a simplistic application of the interactive approach to VRP solving. The appeal of the method is its possibility of combining extensive and long computations done with a computer with the intuition and judgement of the human mind. Krolak et al. [29] have conducted research in this direction.

Problem Number (1)	Number of Nodes	Number of Depots	Load Capacity	Algorithm I Total Distance	Algorithm I CPU Time(2) in sec	Sweep Algorithm Total Distance (5)	Sweep Algorithm CPU Time(2)(5) in sec
1	50	4	80	604.60	2.04(4)	593.16	9.34
2	50	4	160	508.07(3)	9.08	486.19	13.23
3	100	2	100	1124.03	3.15	1066.65	51.24
4	100	2	200	845.03	3.18	778.87	227.67
5	100	4	100	959.64	6.16	939.45	52.48
6	100	2	100	901.83	3.15		

(1) Problems 1-5 are given by Gillett and Johnson [18] (in [18], #6, 7, 20, 21, 23).

(2) On IBM 370/168.

(3) 10 trials have been attempted with different values of γ, the best solution is given and CPU time is total time.

(4) No grid has been used for this case.

(5) Figures reported in [18] on and IBM 370/168.

Table VI Computations with the Proposed Algorithm.

VII. ALGORITHM II: EXTENSION OF ALGORITHM I FOR LARGE PROBLEMS

The multi-depot VRP can be viewed as a two-step process: first, nodes have to be allocated to depots; then routes are built which link nodes assigned to the same depot. Ideally, it is most efficient to deal with the two steps simultaneously as in Algorithm I. When faced with larger problems, with say 1000 nodes, this method is no longer tractable computationally, and we might try to divide the problem into as many subproblems as there are depots and to solve each subproblem separately.

Algorithm II attempts to implement this philosophy while using the ratios r(i) introduced by Gillett and Johnson [18]. If a given node i is much closer to one depot than any other depot, i will be served from its closest depot. When node i is equidistant from several depots, the assignment of i to a depot becomes more difficult. For every node i, we determine the closest depot k_1 and the second closest depot k_2. If the ratio $r(i) = d_i^{k_1}/d_i^{k_2}$ is less than a certain chosen paramenter δ $(0 \leq \delta \leq 1)$, we assign i to k_1; in the case $r(i) \geq \delta$ we say that node i is a border node.

Clearly, if $\delta = 0$, all nodes are declared border nodes and if $\delta = 1$, all nodes are assigned to their closest depot. For a given problem, we can fix the number of border nodes as we wish, by varying δ.

The method proceeds as follows. In the first step, we ignore the nonborder nodes and Algorithm I is applied to the set of border nodes. The algorithm allocates the border nodes to depots and simultaneously builds segments of routes connecting these nodes. At the end of this first step, all nodes of our problem are assigned to some depot and all border nodes are linked on some routes. The solution to the VRP is produced depot by depot using single depot VRP techniques. The segments of routes which are built on border nodes are extended to the remaining nodes.

It is felt that the efficiency of the method depends on how many border nodes Algorithm I can handle. One approach which allows a maximal number of border nodes to be considered involves the ordering of nodes by decreasing ratios r(i). Alternatively, one can experiment with a list of increasing values of δ. As δ increases from 0 to 1 the number of border nodes decreases. The method described here has been implemented with real data taken from the distribution information of a local newspaper. The problem, with almost 600 nodes and 2 depots, ran in under 55 seconds.

VIII. POST-PROCESSOR

In a previous section we mentioned the possibility of improving the solution obtained by Algorithm I by modifying the routes it produced. In this section, we discuss a computerized procedure to perform this task.

We suppose that initially we are given a solution to the VRP. An arbitrary orientation is assigned to each route so that for each node i, we can define pr(i) to be the node or depot which precedes i on its route and fl(i) to be the node or depot which follows i on its route.

If a node j is inserted between nodes i and fl(i), the reduction in distance traveled can be computed as:

$$u_{ij} = d_{pr(j),j} + d_{j,fl(j)} + d_{i,fl(i)} - d_{pr(j),fl(j)} - d_{i,j} - d_{j,fl(i)}.$$

To any pair of nodes i and j, we can consider the savings u_{ij} corresponding to the insertion of node j between i and fl(i). In addition, if node i is the first node served by a route, then the savings v_{ij} associated with the insertion of node j between i and pr(i) has to be taken into account. If the number of routes is r, then the possible savings can be stored in a rectangular matrix W of dimension $(N+r) \times N$. The first N rows correspond to savings u_{ij} and the last r rows refer to the savings v_{ij}. We note that in general $w_{ij} \neq w_{ji}$.

The post-processor searches for the largest element of W which is feasible, and performs the indicated insertion. The operation is then repeated until no further reduction in total distance traveled is possible. The details are very much like those of Algorithm I and are not repeated here.

The post-processor uses a savings approach which differs from the Clarke and Wright algorithm in several respects [35]. In general, updating the savings matrix is more cumbersome as many more savings might change in value. Typically, in the Clarke and Wright algorithm, iterations become shorter as the procedure progresses because the number of nodes to consider decreases, whereas in the post-processor, each iteration takes about the same amount of time.

With a code for implementing the ideas discussed, we have tried the approach with problems reported in Table VI. Results are summarized in Table VII.

Our method has produced solutions comparable with the ones reported by Gillett and Johnson while using much less computation time. In an exceptional case the Gillett and Johnson algorithm required 227.67 seconds of CPU time; our solution was slightly higher while using only about 2% of the CPU time.

Problem (Parameters Specified in Table VI)	Algorithm I		Post-Processor			Improved Solution	Sweep Algorithm	
	Solution	CPU Time in sec	Reduction in Distance Traveled	Number of Insertions Performed	CPU Time in sec		Solution	CPU Time in sec
1	604.60	2.04	10.83	3	0.96	593.77	593.16	9.34
2	508.07	9.08	21.42	6	1.14	486.65	486.19	13.23
3	1124.03	3.15	56.55	21	4.26	1067.48	1066.65	51.24
4	845.93	3.18	29.35	9	2.70	815.68	778.87	227.67
5	959.64	6.16	21.35	12	3.09	938.45	939.45	52.48

Table VII Computations with the Post-Processor.

IX. COMPUTERIZED ROUTING OF NEWSPAPER DISTRIBUTION VEHICLES

In the newspaper logistics system where yesterday's product is worthless today, the allocation and efficient routing of vehicles for the purpose of delivering newspapers on a daily basis is of major importance. In this section, we describe briefly a study that we initiated for an urban newspaper.

Medium to large size newspapers must service several hundred delivery points in and around a city, each with specific demands (in quantity of newspapers). Delivery deadlines or earliest delivery time constraints imposed by either the production department of the newspaper or customer requirements are major considerations. In addition, the vehicle fleet may contain different types of trucks with different capacities and there may be several afternoon editions of a newspaper.

During the period January 1974 to June 1975, while working with the American Newspaper Publishers Association, we investigated issues associated with newspaper delivery systems, particularly the use of our single-depot computer code for specifying vehicle routes for bulk deliveries. The overall goal of our preliminary study was to determine if computerized approaches to newspaper vehicle routing and scheduling looked sufficiently promising from the viewpoints of delivery cost reductions and service quality improvements to warrant further, more detailed investigation.

In order to gain an understanding of routing in the real-world environment of newspapers, we studied the delivery system of a local newspaper - The Worcester Telegram, a paper with an evening city circulation of about 92,000. The staff at the Worcester Telegram supplied us with listings of their vehicle routes within the city limits of Worcester, and appropriate information concerning the associated drop points (approximately 600 in number). The data contained such items as numbers and sizes of delivery vehicles, vehicle capacities, present vehicle routes, and newspaper demands per drop point.

After using a detailed map of the geographic area to define a rectangular grid structure, we specified drop points by coordinates keyed to the grid. Vehicle routes that emerge as output from the input data were based, then, on Euclidean distances. This approach seemed to provide a very reasonable estimate for the true travel distances. Routing data may then be transferred to street maps manually by those people most familiar with the particular city.

The algorithm (presented in detail previously) aims to form routes which minimize the total distance traveled by the fleet while meeting demand requirements. As our experience indicates, this objective also tends to yield a small vehicle fleet and tends not to give excessive route times, even when route-time limitations are not explicitly specified by the user.

The computer code allows vehicles to be loaded to the limits of their functional capacities. These capacities are often hazy figures, at best, but approximations suffice, particularly since various capacity assumptions can be tested readily due to the algorithm's rapid performance. Although maximum route times can be imposed upon any route, we have found in our experimentation that these conditions usually are met even when they are not prespecified. When they are not met, simple manual changes can be made to the routing pattern to satisfy these conditions.

The algorithm ignores possible dependence of route patterns upon time when sequencing drop points for each route. In further testing, time schedules as well as routing, must be looked at, and factors that influence schedules, such as delivery priorities and deadlines, traffic patterns, timing of production, and others, must be taken into account.

We began with data from approximately 600 drop points. However, in several cases when geographical locations were indistinguishable in terms of rectangular coordinates we were able to aggregate two small drop points into one larger drop point. Application of our routing algorithm to the Worcester Telegram (W-T) data yielded the following results:

(1) The drop points within the city limits were covered by 13 routes in contrast to the 20 routes used in the W-T solution.

(2) Several of the computer-dictated routes tended to follow present W-T routes; others did not. The routes generated from our algorithm consolidated the W-T routing patterns. The fact that the proposed routes often followed W-T routes was considered reassuring by everyone involved.

(3) A routing concept emerged that was dramatically different from the one in use. Several of the computer-generated routes called for deadheading to and from an outlying area. Although contrary to previous W-T practice, the W-T seemed quite receptive, and even enthusiastic towards these route structures.

(4) In general, vehicles were loaded to more than 50 percent of functional capacity; they averaged 67 percent capacity.

The Circulation Department of the Telegram appeared to be pleased with the results presented. They had invested an appreciable amount of time in gathering data and seemed to feel their investment was indeed worthwhile. As a result of uncovering large amounts of input data early in the study, many hidden aspects of their routing system were brought to light. W-T personnel decided not to wait for our results, in some cases, but rather to examine their own routing policies immediately in a way they had never before. This led to an alteration of several of their routes with an apparent improvement in efficiency. In addition, this self-evaluation resulted in a greater understanding of their delivery systems, not only with regard to routing policies, but in organization of data.

On the basis of our preliminary recommendations several of the proposed routes have been implemented, at least in part. Insofar as our research was of an exploratory nature and dealt primarily with the routing aspect, we feel that our efforts have been quite successful. Our investigation confirmed what we suspected at the outset - namely, there are deep, subtle questions involved and yet substantial improvements can be realized. Since there are so many variables that enter into a sound routing policy, we believe that through a computerized approach this problem can be handled more intelligently than it has been in the past.

X. CONCLUSION

This paper has presented the vehicle routing problem and the multi-depot VRP, discussed various integer programming formulations, studied several heuristic approaches, and introduced very efficient algorithms which have been implemented successfully. The indication is that the suggested procedures are efficient and can be used as effective decision-making tools for large scale vehicle routing problems encountered in practice.

ACKNOWLEDGEMENTS

We are grateful to Professor F. J. Reintjes for his participation in the ANPA funded study at MIT of newspaper delivery and to Mr. Richard Lakus, circulation manager of the Worcester Telegram, for his assistance.

REFERENCES

1. Balinski, M. and R. Quandt, "On an Integer Program for a Delivery Problem," *Operations Research*, Vol. 12, No. 2, 1964, pp. 300-304.

2. Bellmore, M. and S. Hong, "Transformation of Multisalesmen Problem to the Standard Traveling Salesman Problem," *JACM*, Vol. 21, No. 3, July 1974, pp. 500-504.

3. Bellmore, M. and G. Nemhauser, "The Traveling Salesman Problem: A Survey," *Operations Research*, Vol. 16, 1968, pp. 538-558.

4. Beltrami, E. and L. Bodin, "Networks and Vehicle Routing for Municipal Waste Collection," *Networks*, Vol. 4, No. 1, 1974, pp. 65-94.

5. Bennett, B. and D. Gazis, "School Bus Routing by Computer," *Transportation Research*, Vol. 6, No. 4, December 1972, pp. 317-325.

6. Bodin, L., "A Taxonomic Structure for Vehicle Routing and Scheduling Problems," *Comput. and Urban Soc.*, Vol. 1, 1975, pp. 11-29.

7. Cassidy, P. and H. Bennett, "TRAMP - A Multi-depot Vehicle Scheduling System," *Operational Research Quarterly*, Vol. 23, No. 2, pp. 151-163.

8. Christofides, N. and S. Eilon, "An Algorithm for the Vehicle Dispatching Problem," *Operational Research Quarterly*, Vol. 20, 1969, p. 309.

9. Christofides, N. and S. Eilon, "Algorithms for Large Scale TSP's," *Operational Research Quarterly*, Vol. 23, 1972, p. 511.

10. Christofides, N., "The Vehicle Routing Problem," *NATO Conference on Combinatorial Optimization*, July 1974, Paris.

11. Clarke, G. and J. Wright, "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points," *Operations Research*, Vol. 12, No. 4, 1964, pp. 568-581.

12. Dantzig, G. and J. Ramser, "The Truck Dispatching Problem," *Management Science*, October 1959, pp. 81-91.

13. Eilon, S., C. Watson-Gandy and N. Christofides, *Distribution Management*, Griffin, London, 1971.

14. Floyd, R., "Treesort Algorithm 113," *ACM Collected Algorithms*, August 1962.

15. Gabbay, H., "An Overview of Vehicular Scheduling Problems," *Technical Report No. 103*, M.I.T. Operations Research Center, September 1974.

16. Garvin, W., H. Crandall, J. John and R. Spellman, "Applications of Linear Programming in the Oil Industry," *Management Science,* Vol. 3, No. 4, July 1957, pp. 407-430.

17. Gaskell, T., "Bases for Vehicle Fleet Scheduling," *Operational Research Quarterly,* Vol. 18, 1967, p. 281.

18. Gillett, B. and J. Johnson, "Sweep Algorithm for the Multiple Depot Vehicle Dispatch Problem," presented at *The ORSA/TIMS Meeting,* San Juan, Puerto Rico, October 1974.

19. Gillett, B. and L. Miller, "A Heuristic Algorithm for the Vehicle Dispatch Problem," *Operations Research,* Vol. 22, 1974, p. 340.

20. Golden, B., "Vehicle Routing Problems: Formulations and Heuristic Solution Techniques," *Technical Report No. 113,* M.I.T. Operations Research Center, August 1975.

21. Golden, B., T. Magnanti and H. Nguyen, "Implementing Vehicle Routing Algorithms," *Technical Report No. 115,* M.I.T. Operations Research Center, September 1975.

22. Held, M. and R. Karp, "The Traveling-Salesman Problem and Minimum Spanning Trees," *Operations Research,* Vol. 18, No. 6, 1970, pp. 1138-1162.

23. Held, M. and R. Karp, "The Traveling-Salesman Problem and Minimum Spanning Trees, Part II," *Mathematical Programming,* Vol. 1, No. 1, 1971, pp. 6-25.

24. Hudson, J., D. Grossman and D. Marks, "Analysis Models for Solid Waste Collection," *M.I.T. Civil Engineering Report,* September 1973.

25. IBM Corporation, "System 360/Vehicle Scheduling Program Application Description - VSP," *Report H20-0464,* White Plains, New York, 1968.

26. Karg, L. and G. Thompson, "A Heuristic Approach to Solving Traveling Salesman Problems," *Management Science,* Vol. 10, pp. 225-248.

27. Kershenbaum, A. and R. Van Slyke, "Computing Minimum Spanning Trees Efficiently," *Proc. of 1972 ACM Conference,* Boston, August 1972.

28. Klincewicz, J., "The Tyagi Algorithm for Truck Dispatching," *UROP Final Project Report*, M.I.T., 1975.

29. Krolak, P., W. Felts and J. Nelson, "A Man-Machine Approach Toward Solving the Generalized Truck Dispatching Problem," *Transportation Science*, Vol. 6, No. 2, 1972, p. 149.

30. Lin, S., "Computer Solutions of the TSP," *Bell Systems Technical Journal*, Vol. 44, 1965, p. 2245.

31. Lin, S. and B. Kernighan, "An Effective Heuristic Algorithm for the TSP," *Operations Research*, Vol. 21, 1973, p. 498.

32. Little, J. D. C., K. Murty, D. Sweeney and C. Karel, "An Algorithm for the Traveling Salesman Problem," *Operations Research*, Vol. 11, No. 6, 1963, pp. 972-989.

33. Miller, C., A. Tucker and R. Zemlin, "Integer Programming Formulation of Traveling Salesman Problems," *JACM*, 7, 1960, pp. 326-329.

34. Newton, R. and W. Thomas, "Bus Routing in a Multi-School System," *Computers and Operations Research*, Vol. 1, No. 2, 1974, pp. 213-222.

35. Nguyen, H., "Multi-Depot Vehicle Routing Problems," Master's Thesis, Sloan School of Management, M.I.T., 1975.

36. Orloff, C., "A Fundamental Problem in Vehicle Routing," *Networks*, Vol. 4, No. 1, 1974, pp. 35-64.

37. Orloff, C., "Routing a Fleet of M Vehicles to/from a Central Facility," *Networks*, Vol. 4, No. 2, 1974, pp. 147-162.

38. Orloff, C. and D. Caprera, "Reduction and Solution of Large Scale Vehicle Routing Problems," *Technical Report 75/TR-7*, Princeton University, Transportation Program, July 1975.

39. Pierce, J., "Direct Search Algorithms for Truck-Dispatching Problems, Part I," *Transportation Research*, Vol. 3, 1969, pp. 1-42.

40. Robbins, J., J. Shamblin, W. Turner and D. Byrd, "Development of and Computational Experience with a Combination Tour Construction-Tour Improvement Algorithm for Vehicle Routing Problems," presented at *The ORSA/Tims Meeting*, Las Vegas, Nevada, November 1975.

41. Russell, R., "An Effective Heuristic for the M-Tour Traveling Salesman Problem with Some Side Conditions," presented at *The ORSA/TIMS Meeting*, Las Vegas, Nevada, November 1975.

42. Shapiro, D., "Algorithms for the Solution of the Optimal Cost Traveling Salesman Problem," Sc. D. Thesis, Washington University, St. Louis, 1966.

43. Svestka, J. and V. Huckfeldt, "Computational Experience with an M-Salesmen Traveling Salesman Algorithm," *Management Science*, Vol. 19, No. 7, 1973, pp. 790-799.

44. Tillman, F. and T. Cain, "An Upper Bounding Algorithm for the Single and Multiple Terminal Delivery Problem," *Management Science*, Vol. 18, No. 11, 1972, pp. 664-682.

45. Turner, W., P. Ghare and L. Fourds, "Transportation Routing Problem - A Survey," *AIIE Transactions*, Vol. 6, No. 4, December 1974, pp. 288-301.

46. Tyagi, M., "A Practical Method for the Truck Dispatching Problem," *J. of the Operations Research Society of Japan*, Vol. 10, 1968, pp. 76-92.

47. Webb, M., "Relative Performance of Some Sequential Methods of Planning Multiple Delivery Journeys," *Operational Research Quarterly*, Vol. 23, No. 3, 1972, pp. 361-372.

48. Williams, J., "Algorithm 232: Heapsort," *Comm. ACM*, Vol. 7, No. 6, 1964, pp. 347-348.

49. Wren, A. and A. Holliday, "Computer Scheduling of Vehicles from One or More Depots to a Number of Delivery Points," *Operational Research Quarterly*, Vol. 23, 1972, pp. 333-344.

50. Yellow, P., "A Computational Modification to the Savings Method of Vehicle Scheduling," *Operational Research Quarterly*, Vol. 21, 1970, p. 281.

Research of first author was supported in part by the American Newspaper Publishers' Association and by the Department of Transportation, Contract DOT-TSC-1058. B. L. Golden is now at the University of Maryland, College Park, Maryland.

Research of second author was supported in part by the Office of Naval Research, Contract N00014-75-C-0556 and by the Army Research Office, Durham, Contract DAHCO-73-C-0032.

Paper received September 24, 1975.