A Fundamental Problem in Vehicle Routing

C. S. Orloff
Princeton University
Princeton, New Jersey

ABSTRACT

An important but difficult combinatorial problem, in general, is to find the optimal route for a single vehicle on a given network. This paper defines a problem type, called the General Routing Problem, and gives an algorithm for its solution. The classical Traveling Salesman Problem and the Chinese Postman Problem are shown to be special limiting cases of the General Routing Problem. The algorithm provides a unified approach to both node and arc oriented routing problems, and exploits special properties of most real transportation networks such as sparsity of the associated adjacency matrix, and the tendency for arc symmetry at many nodes. For node oriented routing problems, this approach tends to produce large reduction in effective problem size.

I. INTRODUCTION

An important but difficult combinatorial problem, in general, is to find the optimal route for a single vehicle. Vehicle routing problems are usually stated either in terms of a set of required stops on a network which must be visited, or a set of required arcs on a network which must be traversed. The problem of visiting all nodes of a network in minimum cost is the classical Traveling Salesman Problem (henceforth abbreviated TSP). The problem of traversing all arcs of a network in minimum cost is the well known Chinese Postman Problem (abbreviated CPP). The problem of traversing each arc (at least once) in a required subset of all the arcs of a network in minimum cost is called, in this paper, the Rural Postman Problem (abbreviated RPP). The General Routing Problem (abbreviated GRP) is to find the minimum

Networks, 4: 35-64

cost tour which traverses each arc in a required subset of arcs of the network and which visits each node in a required subset of nodes of the network. Clearly, the TSP, CPP, and RPP are all special cases of the GRP.

In this paper, the terms network and graph are used synonymously. In a transportation network, arcs correspond to streets or parts of streets and nodes correspond to street intersections and locations on a street. For our purposes, it will be useful to distinguish between two types of nodes. Nodes that represent actual street intersections on the transport network are called junctions. Nodes that represent required vehicle visits (pickup or delivery points) are called stops. We note that all stops which are not also junctions are nodes of degree two (they have two and only two incident arcs). We will be concerned mainly with undirected and directed networks in this paper. Networks with some one-way and some two-way streets are called mixed. Section VII deals with mixed networks. Unless explicitly stated otherwise, the terms network and graph refer to the undirected case.

We sometimes denote a chain from node i to node j by $i \rightarrow j$. When graph $G = (N;A)$ is undirected, we use the terms chain and path synonymously. If $i = j$ then we call the $i \rightarrow j$ chain a cycle. Requiring a vehicle route to be a cycle rather than an origin→destination chain is not restricting because we can always equivalently look for cycles of the form origin→destination plus a (destination, origin) arc.

On a complete graph each node is incident to $|N|-1$ arcs (we say that the node has degree $|N|-1$). For vehicle routing problems, transport network $G = (N;A)$ is usually not complete. We often want to work on a surrogate complete graph which is equivalent (for routing purposes) to the original transport network. Given transport network $G = (N;A)$, we denote by $F(N_G)$ the complete graph on node set N derived from network G. The cost of arc (i,j) in $F(N_G)$ equals the cost of traversing a least cost $i \rightarrow j$ chain in G, if $i \neq j$. If $i = j$ the cost of arc (i,j) is infinity (which insures that the vehicle routes actually go somewhere). Calculating the set of least cost chains connecting all pairs of nodes in G is easily done in $|N|^3$ simple operations [10]. For any transport network this need only be done once. Notice that on $F(N_G)$, it is no longer possible to distinguish between original junction nodes on G, and non-junction nodes.

The property of traversability on a graph has traditionally been closely associated with the existence of Hamiltonian cycles and/or Euler tours. Given a graph $G = (N;A)$ a Hamiltonian cycle

is a cycle which visits every node in N once and only once. An Euler tour is a cycle which traverses every arc in A once and only once. In node oriented routing problems, such as parcel-delivery or mail-pickup routing, demand for service is better represented at a point (node) on the graph. In arc-oriented routing problems, such as street cleaning, garbage pickup, and mail delivery, demand for service is better represented as distributed along an arc. When demand is probabilistic, as opposed to fixed, it is usually easier (statistically) to estimate the aggregate demand on an arc, than at each discrete point on the arc.

In this paper, node oriented routing and arc oriented routing are seen to be special cases of general routing. It is also shown that for most real vehicle routing problems, the nature of traversability is more efficiently represented by the Euler tour than by the Hamiltonian cycle. This is because, regardless of the distribution of demand points, vehicles are restricted to a fixed, somewhat limited network (compared to the fully connected network).

Furthermore, Euler tours are quite simply characterized, and easily identified, while no useful, elegant characterization of Hamiltonian cycles is known. Any Hamiltonian cycle on a graph G can be viewed as an Euler tour on a subgraph of G. This paper describes how to identify and construct this subgraph of G and how to connect it to other cycles in G which must be traversed.

II. A REVIEW OF ROUTING

1. The Traveling Salesman Problem

Given the graph G = (N;A) and a non-negative inter-node cost matrix C, the TSP concerns finding the minimum cost cycle in G that visits every node at least once. If only a subset of the nodes are required to be visited, then let G be the connected subgraph corresponding to that subset. Clearly, G must be connected for such a cycle to exist. We denote the cost of traversing arc (i,j) ε A by C_{ij}. It is well known that if C satisfies the triangle inequality ($C_{ij} + C_{jk} \geq C_{ik}$ for all i,j and k), there is an optimal tour in which each node is visited once and only once. As Bellmore and Nemhauser [2] state, "This result is important because almost all algorithms for the TSP are designed to find the minimal length tour that goes through each node exactly once." If C does not satisfy the triangle inequality, we replace it with C', where C'_{ij} is the cost of the

least cost path or chain (of arcs) from node i to node j on graph G. So the TSP on G is generally solved by finding the optimal sequence of nodes to be visited on $F_N(G)$ and assume that each node is connected to the next node on the tour by the least cost chain. In other words, using this transformation from C to C', we are really converting the TSP on G to a problem of finding the minimum cost Hamiltonian tour on $F(N_G)$. When the routing concerns buses on an undirected graph, this least cost chain assumption may force the bus to reverse direction in mid-street upon completing a stop. In some cases this may not be desirable, or even possible, and thereby cause the tour to be infeasible.

Assume C satisfies the triangle inequality, and that G is connected. Then the TSP can be written more precisely as an integer program as follows:

$$\text{Minimize} \qquad Z = \sum_{i=1}^{|N|} \sum_{j=1}^{|N|} C_{ij} X_{ij} \tag{1}$$

$$\text{Subject to} \quad \sum_{i=1}^{|N|} X_{ij} = \sum_{i=1}^{|N|} X_{ji} = 1 \qquad j = 1, 2, \ldots, |N| \tag{2}$$

$$\sum_{i \in N^*} \sum_{j \in N^*} X_{ij} \leq |N^*| - 1 \qquad \text{for all proper subsets of nodes } N^* \subset N, \ |N^*| \neq 0 \tag{3}$$

$$X_{ij} = \begin{cases} 1 & \text{if node j directly follows node i on the sequence of tour stops} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where we set $C_{ii} = \infty \ \forall \ i$ to insure $X_{ii} = 0 \ \forall \ i$ and where $|N^*|$ denotes the cardinality of set N^*.

Equation 2 constrains the tour to enter and leave every node. Equation 3 simply states that no subtour can be a feasible part of a tour (e.g., that the route must be made with a single vehicle). A subtour is actually defined by Equation 3 to be a cycle with fewer than $|N|$ nodes.

At present there exists no efficient algorithm for solving the general TSP, and recent research by Karp [11] gives convincing evidence that the TSP is inherently complex computationally; that a polynomial bounded algorithm does not exist for the TSP.

Nevertheless, there are real routing problems like the TSP which must be solved, and we would like to exploit the special structure of transportation networks to solve them as efficiently as possible. There do exist some efficient heuristic methods [2][15], and some exact methods which are efficient only for very small problems [2], but for larger problems, the TSP subtour-elimination approach has provided one of the few relatively successful exact methods [1].

We can see that embedded in the TSP, in Equations 1, 2, and 4, is an assignment problem (abbreviated AP), which can be easily solved with an upper bound on the number of steps (labelings) of $1/2(n^2+3n-2)$, where n is the number of nodes [6]. The TSP subtour-elimination approach solves the TSP by first solving the AP associated with it (thereby ignoring constraint 3). If the optimal solution to the AP satisfies constraint 3, then it must be the optimal solution to the TSP. If the AP solution does not satisfy this constraint, then a subtour exists and it must be eliminated. For example, if $|N| = 5$, then the solution $X_{14} = X_{45} = X_{51} = X_{23} = X_{32} = 1$ is feasible to the AP, but not to the TSP since it contains two subtours. The subtour is eliminated by adding a constraint to the associated AP which makes certain subtours infeasible, but which eliminates no feasible TSP solutions. The added constraint makes the current AP solution infeasible, and the modified AP is resolved. The modified AP can be easily solved because a good dual feasible solution is available from the previous AP. If S is the set of nodes comprising a subtour in the current AP solution, the constraint

$$\sum_{i \in S} \sum_{j \in \bar{S}} X_{ij} \geq 1 \qquad \text{where } \bar{S} = N - S$$

breaks that subtour. This constraint can be added in such a way as to leave the nice properties of the AP intact, which permit it to be resolved quickly. In short, the constraint is imposed by partitioning the solution space of the AP into $|S|$ subspaces which are collectively exhaustive. Each subspace corresponds to a node of the subtour being forced to assign to a node in $\bar{S}$. The TSP is decomposed into $|S|$ assignment problems, one for each subspace, which are then solved. If subtours remain, this branch and bound procedure continues. All branches of the tree representing the partitioned solution space are followed until each one produces either a feasible TSP solution, or is bounded.

Several different methods exist for partitioning the solution space to break subtours [1] [7]. The structure of the AP is preserved because the partitioning is done solely by setting

certain arc costs equal to infinity. For example, if node i is in subtour S, it can be forced out of S by setting $C_{ij} = \infty \ \forall\ j \in S$, and leaving all other costs unchanged. This modified cost matrix represents one of the $|S|$ partitions.

At this point it is useful to clarify the relationship between the AP and the TSP.

Definition: *Let problem P be: minimize C(X) s.t. $X \in A$.*
Let problem R be: minimize C(X) s.t. $X \in B$.
Then R is called a relaxation of P if $B \supseteq A$.

Theorem 1: *Let X_R^* be the optimal solution to problem R with cost $C(X_R^*)$.*
Let X_P^* be the optimal solution to problem P with cost $C(X_P^*)$.
Then $C(X_R^*) \leq C(X_P^*)$.

Proof: $X_P^* \in A$ and $A \subseteq B \Rightarrow X_P^* \in B$ while X_R^* by assumption is the minimum cost solution in B.

Since the AP is a relaxation of the TSP, its solution provides a lower bound on the optimal TSP solution at each node of the branching tree.

Bellmore and Malone have shown that this subtour elimination algorithm works best for directed networks, where the cost matrix C is asymmetric. They have also shown that this subtour-elimination procedure works poorly for symmetric matrices C which satisfy the triangle inequality. This is because a large amount of work is required to eliminate all the many subtours containing only 2 and 3 nodes. Since $C_{ij} = C_{ji}$ for symmetric C, subtours of the form $i \to j \to i$ are very attractive. Since $C_{ij} + C_{jk} \geq C_{ik} = C_{ki}$, C_{ki} is not independent of $C_{ij} + C_{jk}$. The three variables are positively correlated, consequently subtours of the form $i \to j \to k \to i$ have a tendency to occur. To bypass the problem of size 2 subtours, the TSP on undirected networks can be reformulated as a 2-matching problem subject to an additional no subtour constraint. In the 2-matching problem, only X_{ij} or X_{ji} exist, but not both, so size 2 subtours are impossible (although larger subtours are still possible). Just as the previous TSP formulation had an embedded assignment problem, the new TSP formulation has an embedded 2-matching problem, which is

related to the AP. In fact, the AP is a relaxation of the 2-matching problem, which is a relaxation of the TSP.† The general b-matching problem on node set N can be stated as:

$$\text{Minimize} \quad \sum_{i=1}^{|N|-1} \sum_{j=i+1}^{|N|} C_{ij} X_{ij}$$

$$\text{Subject to} \quad \sum_{i=1}^{j-1} X_{ij} + \sum_{i=j+1}^{N} X_{ji} = b_j \quad \forall\, j$$

$$X_{ij} = \begin{cases} 1 & \text{if node i and node j match} \\ 0 & \text{otherwise} \end{cases}$$

X_{ij} exists only for $j > i$.

When $b_j = 2$ for all j, the above problem is called the 2-matching problem (abbreviated 2-MP). The b-matching problem on graph $G = (N;A)$ concerns finding the minimum cost subset M of arcs in A such that each node $j \in N$ is incident to exactly b_j arcs in M (node j has degree b_j on the subgraph corresponding to M). Edmonds [12] has developed a general b-matching algorithm, where the upper bound on the number of computations for an optimal solution is $|N|^4$, independent of b. Considerable computational experience has shown this algorithm to be efficient.

Using the 2-matching formulation in the TSP subtour elimination procedure, first the embedded 2-MP is solved.‡ If no subtours exist, then the solution is optimal to the TSP. If subtours exist, they are eliminated by the branch and bound-like procedure described previously. The subtour-elimination algorithm remains exactly the same, except that the bounding subproblem is the embedded 2-matching problem, where it was previously the embedded AP. The generalized subtour-elimination algorithm is given more explicitly in Section IV.

---

†*Given graph $G = (N;A)$, the AP is a 1-matching problem on equivalent bipartite graph $G' = (N,N;A')$ where A' represents the arcs of $F(N_G)$, but this corresponds to 2-matching (with the addition of size 2 subtour solutions) on the original graph G.*

‡*We are assuming in this paper that C satisfies the triangle inequality, so the matching problem can be solved on G. Otherwise the matching problem would have to be solved on $F(N_G)$.*

2. Checking Solutions for Subtours

An important point in the procedure used to check solutions for subtours is that subtours on $F(N_G)$ do not necessarily correspond to subtours on G, because the arcs in surrogate graph $F(N_G)$ correspond to chains in G. So when the solution to the AP or 2-MP on $F(N_G)$ is checked for subtours, the checking must be done on the corresponding solution on G, not $F(N_G)$. We distinguish between the three following types of solutions to the AP or 2-MP on $F(N_G)$:

1) Disconnected solutions: subtours on $F(N_G)$ correspond to subtours in the corresponding solution on G.
2) Actively connected solutions: no subtours on $F(N_G)$ and therefore no subtours on G.
3) Passively connected solutions: subtours on $F(N_G)$ correspond to a connected tour in the corresponding solution on G.

The optimal tour, for type 3 solutions obviously contains a node that is visited more than once. Let us call this node k, and assume the solution to the AP or 2-MP on $F(N_G)$ contains two subtours S and $\bar{S}$. If the solution is passively connected, and $k \in \bar{S}$, then the solution contains an assigned (i,j) arc in S on $F(N_G)$ which corresponds to an $i \rightarrow k \rightarrow j$ chain on G which connects S and $\bar{S}$ on G.

An illustration will clarify these solution types. Consider the TSP on graph G = (N;A) shown in Figure 1 (with arc costs as shown). Figure 2 shows $F(N_G)$ and Figure 3 gives the internode cost matrix for $F(N_G)$. Solving the AP on this cost matrix we get the solution on $F(N_G)$ shown in Figure 4a. We see that it is a disconnected solution; it contains two subtours. Figure 4b shows the solution on G corresponding to the solution on $F(N_G)$ shown in Figure 4a. From Figures 4a and 4b we see that this type 1 solution, disconnected on both $F(N_G)$ and G. Node 6 is then forced out of the subtour by making the cost of arc (6,5) equal to infinity ($C_{65} = \infty$). Figure 5a shows another AP solution on $F(N_G)$ with the modified cost matrix (so that $C'_{65} = 25$) containing two subtours. This is a passively connected (type 3)

solution because Figure 5b shows that the corresponding solution on G is connected. Figure 6a shows another AP solution on $F(N_G)$. This is a connected solution (type 2), so the corresponding solution on G must also be connected, as seen in Figure 6b.

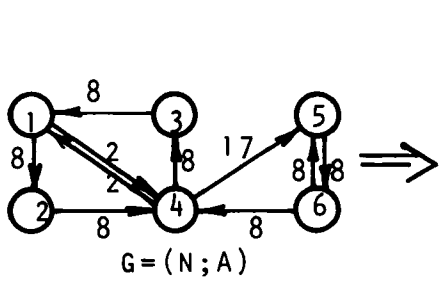


Fig. 1

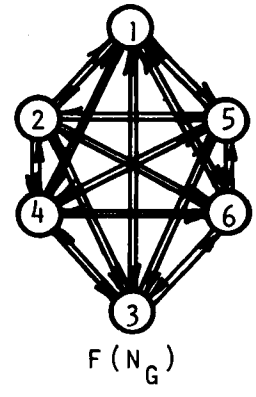
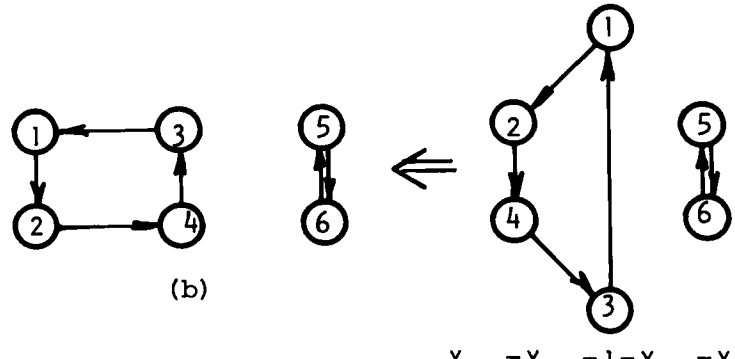


Fig. 2

C' =

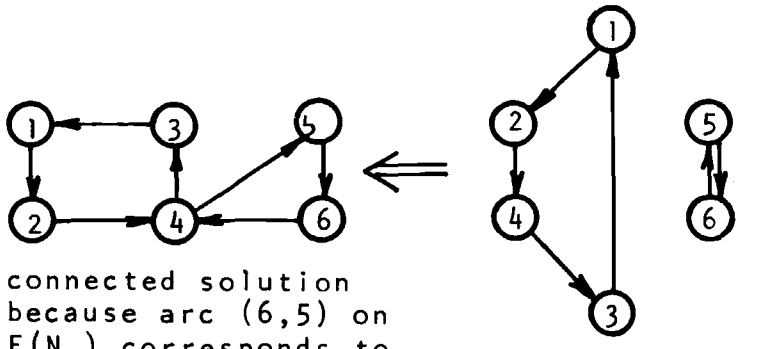
Node	1	2	3	4	5	6
1	∞	8	10	2	19	27
2	10	∞	16	8	25	33
3	8	16	∞	10	27	35
4	2	10	8	∞	17	25
5	18	26	24	16	∞	8
6	10	18	16	8	8	∞

Fig. 3



$X_{56}=X_{65}=1=X_{12}=X_{24}=X_{43}=X_{31}$

Fig. 4



connected solution because arc (6,5) on $F(N_G)$ corresponds to chain 6→4→5 on G

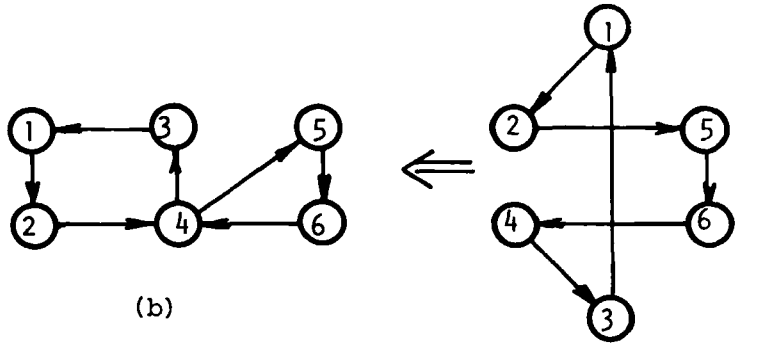
(b)

$X_{12}=X_{24}=X_{43}=X_{31}=1$

$X_{56}=X_{65}=1$

(a)

Fig. 5



(b)

$X_{12}=X_{25}=X_{56}=X_{64}=X_{43}=X_{31}=1$

(a)

Fig. 6

Actually, if the solution on $F(N_G)$ contains subtours, we do not always have to check the corresponding route on G to distinguish between a type 1 and a type 3 solution, because of the following theorem.

Theorem 2: If solution X to the TSP relaxation problem (call it R) on $F(N_G)$ is passively connected, then there exists alternate optima for R.*

Proof: R is a relaxation of the TSP. From Theorem 1, cost (X*) $\leq$ cost (optimal TSP solution). By assumption, X* passively connected implies that there exists a feasible TSP solution on G

with value = cost (X*). For every feasible TSP solution on G there exists an equivalent actively connected TSP solution on $F(N_G)$. Thus another solution X** to R on $F(N_G)$ must exist, which is actively connected, and such that cost (X**) = cost (X*).

The implication of this theorem is that if a solution to R on $F(N_G)$ contains subtours but no alternate optima, it must be a type 1 solution. If alternate optima do exist, then they can all be checked, or alternatively, the corresponding solutions on G can be checked for subtours. The optimal routing algorithms known to the author, which operate on the surrogate complete graph, check only for active connectivity, and thus they may spend time eliminating subtours on F which do not exist on G.

Using the structure of G (shown in Figure 1), this TSP can be reduced to a triviality. For example, requiring node 3 to be visited means that arcs (4,3) and (3,1) must be traversed. Similarly, by deduction, arcs (1,2), (2,4), (5,6), (6,4), and (4,5) must be traversed in any feasible TSP solution. Figure 6b shows the subgraph of G corresponding to all the arcs which must be traversed in any feasible Traveling Salesman Problem solution. Obviously, the best any TSP solution can do is to traverse only this set of arcs we know to be required. Since the subgraph in Figure 6b contains an Euler tour which visits every node in N, this Euler tour must be the optimal TSP solution on G. In a later section these reductions are formalized.

3. The Chinese Postman Problem

The Chinese Postman Problem (abbreviated CPP) is the problem of finding the minimum cost tour which traverses each arc of a given graph at least once. The CPP is closely related to the Euler tour property. An Euler tour is a cycle which traverses each arc of a graph once and only once. If an Euler tour exists for a graph G, then it must be the optimal solution to the CPP on G.

Once it is known that an Euler tour on G exists, it is a simple matter to find one [5]. Several different Euler tours may exist on G. The necessary and sufficient conditions for the existence of an Euler tour on a graph are given by the following theorem:

Theorem 3: [6] *The mixed graph G = (N;A) contains an Euler tour if and only if: a) G is connected; b) every node of G is incident with an even number of arcs; c) for every* $X \subseteq N$,

the difference between the number of directed arcs from X to $\bar{X}$ and the number of directed arcs from $\bar{X}$ to X is less than or equal to the number of undirected arcs joining X and $\bar{X}$.

We note that for completely undirected graphs, condition c is satisfied automatically by condition b. So, only conditions a and b need be checked, which is trivial. For completely directed graphs, condition c requires node symmetry; that the number of arcs directed into a node equal the number of arcs directed out of that node. This condition is also quite easy to verify. For mixed graphs, the easiest way to check condition c is to solve an equivalent network flow problem [6]. Since solving the CPP for mixed graphs is, in general, extremely difficult, while solving the CPP for completely directed or undirected graphs is rather simple, we will not concern ourselves with mixed graphs until Section VII.

To solve the CPP on connected graph G = (N;A), one first checks if an Euler tour exists. If one exists, then it must be the optimal solution to the CPP. If an Euler tour does not exist, then some nodes of G are of odd degree (if G is undirected) or some nodes are asymmetric (if G is a digraph).

Assume that G is an undirected graph and that an Euler tour does not exist on G. Then, some nodes of G are of odd degree. Let E be the set of odd nodes of G. We note that $|E| \equiv 0 \bmod(2)$ since there are always an even number of odd nodes in any graph. The problem of solving the CPP on G now becomes a problem of modifying (augmenting) G in a minimal cost way, so that the resulting graph G' has an Euler tour which is the optimal solution to the CPP on G. This modification is done by adding extra copies of some arcs of A to the graph G to get augmented graph G', so that all nodes of G' are of even degree. By finding the minimum cost set of duplicate arcs, the Euler tour on G' is the CPP solution on G. Since only the nodes in E violate the Euler tour condition, the added arcs must make all nodes of E of even degree while maintaining the parity of all nodes in $\bar{E}$.

To do this we first find the minimum cost chain in G connecting each pair of nodes in E. In other words, we construct $F(E_G)$. Then we do a minimum cost 1-matching on $F(E_G)$. The cost of matching two nodes of E, say i and j, is C'_{ij}, which equals the cost of the least cost chain on G connecting nodes i and j. If K is the set of least cost chains connecting all pairs of nodes in E, then the minimum cost 1-matching problem is to find a set $M \subset K$ of $|E|/2$ chains such that each node in E is the end point of exactly one chain in M, and the sum of the costs associated with all the arcs of M is minimum.

The arcs contained in the chains of M are precisely the arcs which will be duplicated and added to G to form G'. This is because each node b ε E is of odd degree d(b) on G, and is incident to the end of exactly one chain in the optimal 1-matching solution. If node b is contained in no other chains of M, then it has even degree (d(b)+1) in G'. If node b is contained in any other chain of M, then it must be an interior node (not an end node) of the chain. Any interior node of a chain is incident to two arcs in the chain, and thus its degree is increased by two in G'. By the same argument, all nodes in $\bar{E}$ retain their parity.

So, solving the CPP on G becomes a three step process of:

1) Check if G satisfies the Euler tour conditions. If so, find the Euler tour; this is the CPP solution. If not, find the set E of odd nodes of G and go to Step 2.
2) Construct $F(E_G)$. Let C'_{ij} be the cost of the least cost chain on G connecting i ε E and j ε E. Let $C'_{ii} = \infty$ for all i ε E.
3) Solve the minimum cost 1-matching problem on $F(E_G)$. The optimal solution is a set M of chains connecting $|E|/2$ pairs of nodes. Form G' by adding a copy of each arc contained in the set (of chains) M to graph G. G' has an Euler tour by construction. Find an Euler tour. It is the optimal CPP solution on G.

The minimum cost 1-matching problem is the general b-matching problem given previously where $b_j = 1$ for all j, and this is easily solved by Edmonds' b-matching algorithm. We form set M using the following rule: for each X_{ij} or $X_{ji} = 1$, in the optimal solution to the 1-matching problem, add a copy of all the arcs in the least cost chain connecting node i to node j on graph G.

For solving the CPP on fully directed connected graph G = (N;A), a simpler algorithm exists. A minimum cost network flow problem or an equivalent transportation problem is solved, and the solution identifies the arcs of A to be duplicated and added to G to form the augmented graph G' which is symmetric at all nodes. In particular, let E be the set of asymmetric nodes violating the Euler tour conditions on G. Construct $F(E_G)$ and its associated cost matrix C'. For each node b ε E let d(b) be the number of arcs directed into node b minus the number of arcs directed out of node b. Then the transportation problem is:

$$\text{Minimize} \quad \sum_{j=1}^{|E|} \sum_{i=1}^{|E|} C'_{ij} X_{ij}$$

$$\text{Subject to} \quad \sum_{i=1}^{|E|} X_{ij} = d(j) \qquad \forall\, j \ni d(j) < 0$$

$$\sum_{j=1}^{|E|} X_{ij} = d(i) \qquad \forall\, i \ni d(i) > 0$$

$$X_{ij} = \begin{cases} 1 & \text{if nodes i and j match via } i \to j \\ 0 & \text{otherwise.} \end{cases}$$

For each $X_{ij} = 1$ in the optimal solution to the above problem, add a copy of all the arcs in the least cost $i \to j$ chain on G to form G'.

The CPP on digraph G can also be thought of as an equivalent network flow problem of establishing a minimum cost circulation on G subject to lower bounded arc flow. The circulation, which must be symmetric, defines the duplicate arcs to be added to G to obtain G'.

Again, we note that for the CPP, as for the TSP, the matching problem is solved on the surrogate complete graph F, rather than the original network G. For the CPP at least, the structure of G is somewhat exploited since the matching is only on the set of odd nodes. Edmonds and Johnson [5] further this exploitation by giving an extension of the matching algorithm to solve the CPP directly on G.

III. THE RURAL POSTMAN PROBLEM

The problem of finding the minimum cost tour which traverses each arc in a required subset of the arcs of a graph is called (in this paper) the Rural Postman Problem (abbreviated RPP). Consider the undirected graph $G = (N;A)$. Let $R \subseteq A$ be the set of required arcs to be traversed. Let N_R be the set of nodes incident to any arc in R. Consider the subgraph $G_* = (N_R;R)$.

If G_* is 1) connected and 2) all nodes $j \in N_R$ are of even degree, then an Euler tour exists on G_*, and it is the optimal solution to the RPP on G. Let E_* be the set of odd

degree nodes in N_R. If G_* is connected but $|E_*| > 0$, then we solve the RPP by adding copies of arcs of G to G_* in a least cost way to create the augmented graph G'_*. We do this modification in such a way that an Euler tour exists on G'_* that is the optimal solution to the RPP on G.

To be more precise, first we construct $F((E_*)_G)$ by finding the least cost set of chains on G connecting all pairs of nodes in E_*. We note that the least cost $i \to j$ chain may contain arcs not in R (in fact, it will often contain only arcs in $\bar{R}$), because we find these chains on the original graph G. We then solve the minimum cost 1-matching problem on the odd nodes in E_*, which will indicate which $i \to j$ chains will have their arcs copied onto G_* to create the augmented graph G'_*. Since G'_* satisfies conditions 1 and 2 by construction, an Euler tour exists on G'_*. It is the optimal solution to the RPP on G because, by definition, G'_* is the minimum cost such construction from G.

What happens when G_* is not connected? If $|E_*| > 0$, then we must still do a minimum cost 1-matching on E_*, but now this matching is subject to the constraint that the resulting graph G'_* must be connected to satisfy the Euler tour condition 1. A disconnected component $S = (N_S; A_S)$ of G_* is a subgraph of G_* such that S contains no disconnected components and there exists no arc in R with incident nodes $i \in N_S$ and $j \in N_R - N_S$. What happens when one of the disconnected components, call it S_0, of G_* has all even nodes? We must still duplicate some arcs of G and add them to G_*, to connect this component to the rest of the graph in its modified form G'_*. Any one node in S_0 is eligible to be a connection point to the rest of the graph but all nodes in S_0 must have their degree parity unaltered, so the connection will be by adding two arcs of G incident to the same node in S_0 when creating G'_*. We note that an Euler tour exists on S_0, and we call it a subtour.

Again we have a minimum cost matching problem subject to a no-subtour constraint, just as we had for the TSP. But in this case the matching problem requires 1-matching and the no-subtour

constraint is forced by the connectivity requirement of an Euler tour. Clearly a subtour elimination algorithm can be used to solve the RPP, using the 1-MP on E_* as the bounding subproblem. Adding the no-subtour constraint is implemented by exactly the same solution set partitioning procedure used for the TSP.

In some special cases, when a disconnected component has all even nodes (a subtour), the partitioning process can be avoided and the branching tree does not enlarge. For example, when the disconnected component S_k is very far from the rest of the graph, compared to the internode distances within S_k, then any node of S_k will be an approximately equivalent connection point. This is also true when the nodes of S_k are symmetric with respect to the nodes of $\bar{S}_k$. Lastly, node $i \in S_k$ is only eligible to be a connection point to $\bar{S}_k$ if there exists an arc $(i,j) \in A$ for $j \in \bar{S}_k$ on G. This obvious statement eliminates many branches in the subtour elimination branching tree for real transportation networks.

Again we note that for the RPP the matching problem is solved on the surrogate complete graph F, rather than the original network G. Although this approach exploits the structure of G somewhat, it is more efficient to do the matching directly on G.

As expected, the RPP on digraph G is even easier to solve, because the matching problem is replaced by the transportation problem on the set of asymmetric nodes E_*. The procedure is directly analogous to the procedure for solving the CPP on directed graphs. On digraphs, small subtours are not usually obtained, and therefore the subtour elimination algorithm works very well.

IV. THE GENERAL ROUTING PROBLEM

A good argument can be made to support the assertion that most real traveling salesman-like routing problems should be stated in terms of a set of required nodes to be visited *and* a set of required arcs to be traversed in a tour of minimum cost on a transport network G. This problem is called the General Routing Problem (GRP). Before formulating the GRP, we assume that as many required nodes as possible have already been transformed into required arcs. These transformations reduce ultimate problem size, and some rules for the "graph reduction" will be given later. Consider the GRP on undirected graph G = (N;A)

Let $R \subseteq A$ be the set of required arcs to be traversed.
Let N_R be the set of nodes incident to any arc in R.
Let Q be the set of required nodes in N.

By assumption, $Q \cap N_R$ is empty.

Let E_* = set of odd degree nodes in N_R, $T = E_* \cup Q$.
Let $S = \{S_1, S_2 \ldots, S_F\}$ the set of diconnected components in $(N_R; R)$.

All nodes $j \in Q$ are disconnected in $G_* = (Q \cup N_R; R)$ but we do not call them subtours. Assume that no subtours exist initially in the subgraph G_*, so all S_i have at least two odd nodes each (if any initial subtours do exist, then break them by the branching/partitioning method described in Section II).

We construct complete graph $F(T_G)$.

Let C'_{ij} = cost of the least cost chain on G connecting $i \in T$ and $j \in T$ $i \neq j$, $C'_{ii} = \infty \; \forall \; i \in T$.

Then the minimum cost subgraph modification problem for the GRP is:

$$\text{Minimize} \quad Z = \sum_{i=1}^{|T|-1} \sum_{j=i+1}^{|T|} C'_{ij} X_{ij}$$

$$\text{Subject to} \quad \sum_{i=1}^{j-1} X_{ij} + \sum_{i=j+1}^{|T|} X_{ji} = 1 \qquad \text{for all } j \in E_*$$

$$\sum_{i=1}^{j-1} X_{ij} + \sum_{i=j+1}^{|T|} X_{ji} = 2 \qquad \text{for all } j \in Q$$

No disconnected subtours in the augmented graph G'_*.
(active or passive connectivity)

$$X_{ij} = 0 \text{ or } 1$$

X_{ij} exists for $j > i$ only.

Note that the relaxation of the above problem, ignoring the no-subtour constraint, is simply a matching problem. We call this a mixed 1,2-matching problem (abbreviated 1,2-MP) because for each node j, b_j equals 1 or 2. Clearly, Edmonds' general b-matching algorithm can solve the embedded 1,2-matching problem in the GRP.

Letting the 1,2-matching problem on $F(T_G)$ be the new bounding subproblem in the branch and bound subtour elimination procedure, we see that the same procedure solves the GRP. In particular, the method of solution space partitioning by arc cost modifications to impose the no-subtour constraint is directly applicable. The GRP (generalized) subtour elimination algorithm consists of:

Step 1: Assume no initial subtours exist. If they do, then break them by partitioning. Create fully connected graph $F(T_G)$ where $T = Q \cup E_*$. Solve the embedded 1,2-matching problem on $F(T_G)$. If no subtours exist, then go to Step 4. If the solution has subtours (no active or passive connectivity), set the current best feasible solution value equal to infinity. Call the smallest subtour S. Go to Step 2.

Step 2: For the branch with minimum lower bound, eliminate S by partitioning the solution space into k subspaces, where k is the number of nodes in S. Do the partitioning by the methods of Section II (modifying arc costs only). Create k successor branches to this branch, one for each subspace. Solve the 1,2-matching problem for each of these subspaces (successor branches).

Step 3: The solutions to the 1,2-matching problem from Step 2 are divided into those with subtours (on G), and those that are feasible to the GRP (without subtours on G). The best solution from the latter group is compared to the current best feasible solution to determine the new best feasible solution. The solutions with subtours but with value greater than (or equal to) the current best feasible solution, are eliminated from consideration. The other solutions with subtours are put into the set of active solutions under consideration. If the minimum cost solution in the set of active solutions has value greater than the current best feasible solution, then the current best feasible solution is optimal, and go to Step 4. Otherwise, the minimum cost active solution is the branch with minimum lower bound. Call the smallest subtour in this solution S. Go to Step 2.

Step 4: Create the augmented graph G'_* in the following way. For each $X_{ij} = 1$ in the 1,2-MP solution corresponding to the current best feasible solution, add a copy of all the

arcs in the least cost i → j chain on G to subgraph $G_* = (Q \cup N_R; R)$. By construction, an Euler tour exists on G_*' and it is the optimal solution to the GRP on the original network G. Find the Euler tour. Terminate.

Notice that when R is empty, the GRP becomes a TSP. When Q is empty the GRP becomes a CPP for R = A, otherwise it becomes a RPP. So it is not surprising that the TSP subtour elimination algorithm using the embedded 2-matching problem, and the GRP subtour elimination algorithm using the embedded 1,2-matching problem, are essentially equivalent. The major difference is that the 2-matching problem is on $|N|$ nodes, while the 1,2-matching problem is on $|E_* \cup Q|$ nodes. We formalize an obvious but important result in the following theorem.

Theorem 4: Given a TSP on G = (N;A), an equivalent GRP can be formulated by converting required nodes to required arcs such that $|E_* \cup Q| \leq |N|$.

Proof: Since the TSP is a special case of the GRP, a non-negative reduction in problem size always exists.

This theorem is important because solution time for solving both the TSP and GRP is proportional to the size of the node set of the associated matching problem. For the TSP on G = (N;A), the traditional TSP subtour elimination algorithm operating on the surrogate complete graph $F(N_G)$ takes no advantage of the special structure of G. For the most real vehicle routing problems, this transformation from G to $F(N_G)$ is a step backwards because:

a) The node-node incidence (adjacency) matrix of G is usually very sparse, whereas the adjacency matrix of $F(N_G)$ is full.

b) All nodes of $F(N_G)$ are of degree $|N|-1$ whereas the nodes of G tend to be symmetric, and of degree 2 or 4 (or 3), especially for road networks of the modified-grid type.

c) The TSP on $F(N_G)$ is the difficult problem of finding the optimal Hamiltonian cycle, whereas the TSP on G is really concerned with finding the best traversable tour which visits every node (an Euler tour on a subgraph of $G^+ = (N;A+A)$).

Using the GRP approach to routing, even for Traveling Salesman Problems, permits the special structure of G to be exploited by converting required nodes to required arcs, and by using the structure of G in determining what Euler tour conditions are not satisfied. In practice, for traveling salesman-like routing problems on real transport networks, the TSP to equivalent GRP conversion often results in extremely large reduction in problem size. This is especially true for micro-routing on streets with a densely distributed point demand for service.

A comment about the initial elimination of subtours is also in order. If a disconnected component S_0 of subgraph $(N_R;R)$ has all even nodes, then it is an initial subtour because an Euler tour exists on it. Since no nodes of S_0 are in the embedded 1,2-matching problem, the subtour may never be eliminated unless it is broken by the solution space partitioning method. It is broken by forcing each node of S_0 to be the connection to a node in $\bar{S}_0$, with a different node being the connection point in each of the $|S_0|$ partitions. Since parity for each node in S_0 must be retained, this is simply a 2-matching requirement for the node in S_0, where the only eligible matches are contained in $\bar{S}_0$. In other words, to break S_0, add a node of S_0 to Q, the set of required nodes, where each node of S_0 augmenting Q defines a new branch or partition of the solution space.

As expected, solving the GRP on directed graphs is more efficient than solving equivalent size problems on undirected graphs. The algorithm is exactly the same as the subtour elimination algorithm previously given, except that the mixed 1,2-matching problem is replaced by the transportation problem given below (assuming no initial subtours).

Let E_* be the set of asymmetric nodes violating the Euler tour condition in subgraph $(N_R;R)$. For each node $b \in E_*$ let d(b) be the number of arcs directed into node b minus the number of arcs directed out of node b. Let $E_1 = \{\text{all } j \in E_* \ni d(j) < 0\}$ and let $E_2 = E_* - E_1$. Construct complete graph $F(T_G)$ where $T = E_* \cup Q$ with associated cost matrix C'. C'_{ij} is the cost of the least cost $i \to j$ chain on G, $i \neq j$; $C'_{ij} = \infty$ for $i = j$.

The transportation problem is simply:

$$\text{Minimize} \quad Z = \sum_{i=1}^{|T|} \sum_{j=1}^{|T|} C'_{ij} X_{ij}$$

$$\text{Subject to} \quad \sum_{i=1}^{|T|} X_{ij} = d(j) \qquad \forall\, j \in E_1$$

$$\sum_{i=1}^{|T|} X_{ij} = 1 \qquad \forall\, j \in Q$$

$$\sum_{j=1}^{|T|} X_{ij} = d(i) \qquad \forall\, i \in E_2$$

$$\sum_{j=1}^{|T|} X_{ij} = 1 \qquad \forall\, i \in Q$$

$$X_{ij} = \begin{cases} 1 & \text{if chain } 1 \to j \text{ is matched} \\ 0 & \text{otherwise.} \end{cases}$$

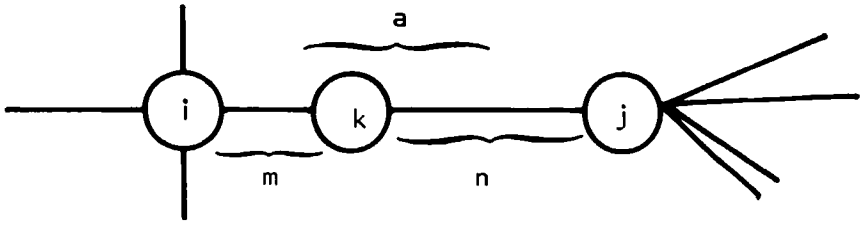
The subtour elimination algorithm also provides sensitivity data. A solution with K subtours gives the problem solver some information as to how well he could do if he had K vehicles. Strong subtours, which tend to persist in altered form after nodes are forced out, generally indicate strong precedence relationships among the nodes on the tour. Within the GRP formulation, the problem solver can replace long segments of the tour that are persistent in heuristic solutions and the strong subtours, by a single required arc. While this may be suboptimal, it drastically reduces problem size.

V. SOLVING TRAVELING SALESMAN-LIKE ROUTING PROBLEMS AS GENERAL ROUTING PROBLEMS

Even when a problem is given only in terms of required nodes, it can usually be transformed into an equivalent problem of required nodes and required arcs, with a reduction in problem size. Any required node, and sometimes many required nodes, can be transformed into one equivalent required arc.

How and when should this required node(s) to required arc transformation be done? Some rules are given below.

Theorem 5: On a connected undirected graph G, let i, j, k, be required stops for a tour. Assume i and j are adjacent junctions of G, and k is anywhere on street (i,j) (see figure below). Assume the non-negative cost function C associated with the arcs of G satisfies the triangle inequality. Then, stops i, j, k, can be replaced by a required arc representing chain i → k → j since it will be in any minimum cost tour visiting the required nodes of G.



Proof: Let a denote arc (i,j) with cost C_a. Let m denote arc segment (i,k) with cost C_m. Let n denote arc segment (k,j) with cost C_n. From the triangle inequality $C_a = C_m + C_n$. Assume arc a is not in an optimal tour. Since any tour is a cycle, and every feasible tour visits all required nodes, let an optimal tour (without any loss of generality) go from k to j via arc segment n, then from j to i via chain x, then from i to j via chain y, then from j to k via arc segment n. Let C_x be the cost of chain x. From optimality, we know $C_x + C_y + 2C_n \leq C_x + C_m + C_n \Rightarrow C_y + C_n \leq C_m$ and $C_x + C_y + 2C_n \leq C_y + C_m + C_n \Rightarrow C_x + C_n \leq C_m$ since (x+m+n) and (y+m+n) are feasible tours. But this violates the triangle inequality. By contradiction, arc a must be in an optimal tour.

If a required node is on or incident to a required arc, it is redundant and can be deleted from the set of required nodes. If vehicles are not permitted to change direction in the middle of a street, any required stop in the middle of a street forces the entire street to be required. If required node k has degree 2 in digraph G, then replace required node k by a required arc representing the two arcs incident to k (assuming the arcs can be traversed in the same direction). An experienced dispatcher can reduce the size of a routing problem by *a priori* making

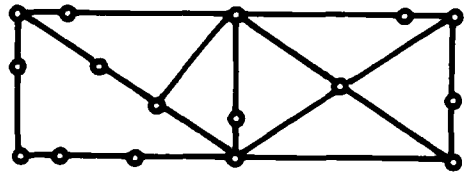
large contiguous segments of a feasible route into a single required arc when he knows that these contiguous segments will be in an optimal solution. An experienced dispatcher, working on a real routing problem on a familiar network, should be able to make considerable reduction in problem size without seriously suboptimizing. In the reduced problem the dispatcher is only concerned with the uncertain parts of the tour.

These reduction rules are obvious, but they have also shown to be quite effective. For other assumptions, such that C satisfies the Euclidean distance measure, other reduction rules can be applied [2]. The required node to required arc conversions are only one source of problem reduction. The other, often more important source, comes from the fact that not all the required arcs contribute to problem size. Only the set of odd nodes (incident to the required arcs) add to problem size.

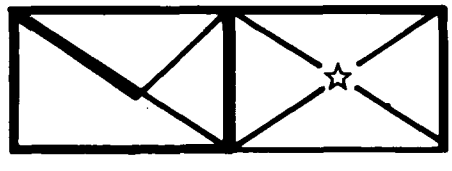
VI. EXAMPLES

In this section simple examples of using the GRP approach to solving TSP-like routing problems are given to illustrate the procedures described in this paper.

A) Consider the TSP on the undirected graph G shown below:

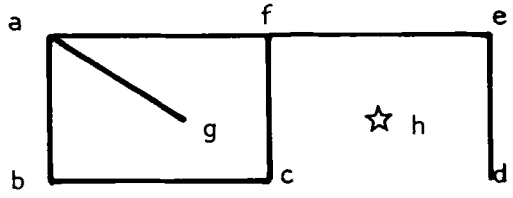


where the small circles shown are the required nodes or stops, and the cost associated with an arc is proportional to its distance shown in true scale above. Using the rules of the previous section we can convert the above TSP to an equivalent GRP shown below,

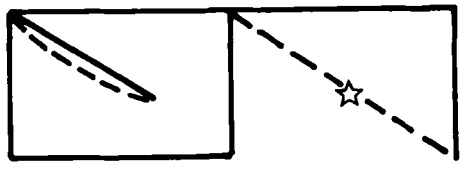


where the heavy lines denote required arcs, and the light lines denote optional arcs. The single required node is denoted by a star.

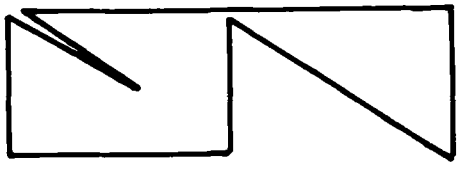
We note that the set of 16 required nodes in the TSP are reduced to a set of 7 required arcs and one required node in the equivalent GRP, and there are only 4 odd nodes on the subgraph of required arcs. The subgraph G_* associated with the set of required arcs and nodes is given below.



We see that G_* contains no disconnected components or subtours, so we simply solve the minimum cost mixed matching problem, requiring 1-matching for all odd nodes a, g, f, d, and 2-matching for required node h. The solution to this matching problem is represented by the dotted lines on G'_* shown below.

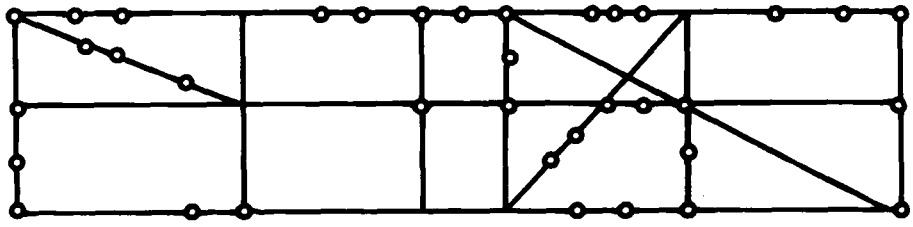


We see that G'_* shown above is connected and has all nodes of even degree, so an Euler tour exists on G'_* and by construction it is the optimal solution to the TSP on G. The Euler tour is simply found and shown below.

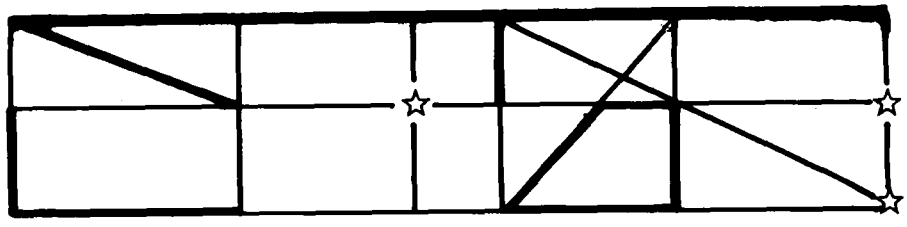


This is the best case, when the TSP can be solved as a GRP without subtour problems. For problems such as garbage truck routing and school bus routing this is often the case. Routing problems on strictly grid networks are also usually quite easy, because all nodes are of degree four (except boundary nodes).

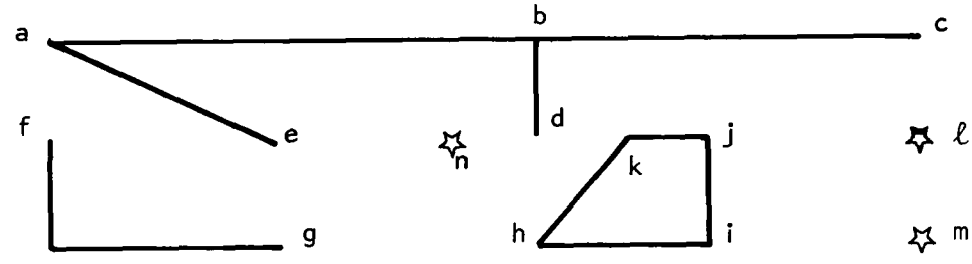
B) Consider the TSP on undirected graph G shown below,



where only the circles shown are the required stops. We can convert this to an equivalent GRP shown below:



where the heavy arcs denote required arcs and the stars denote required nodes. The subgraph G_* associated with the above GRP is given below:



Note that the original TSP on 36 nodes was converted to a GRP with 3 required nodes, 6 odd nodes on the subgraph of required arcs, and one subtour. There are 6 disconnected components

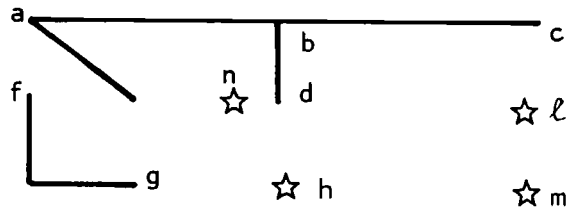
$S_1 = \{a,e,b,d,c\}$ $\quad S_4 = \{n\}$

$S_2 = \{f,g\}$ $\quad S_5 = \{\ell\}$

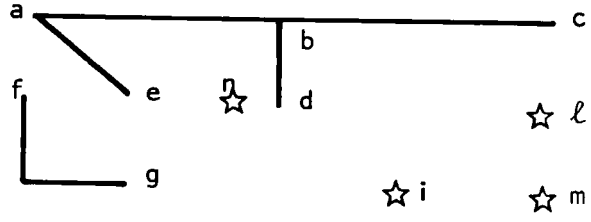
$S_3 = \{h,i,j,k\}$ $\quad S_6 = \{m\}$

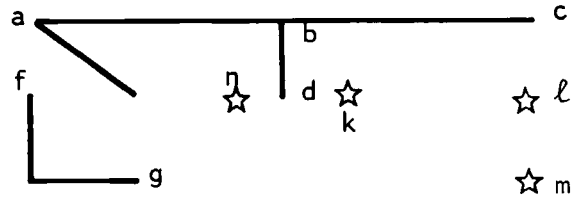
S_3 is a subtour and must be broken. Partioning on S_3 we get the following 4 GRPs.

BRANCH I

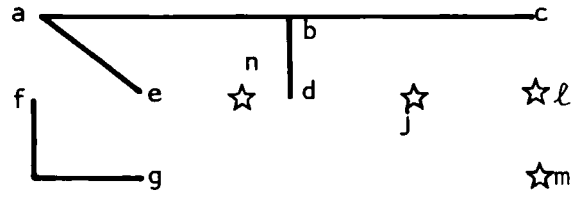


BRANCH II





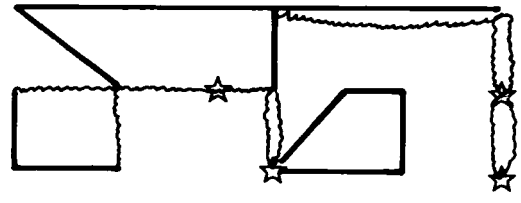
BRANCH III



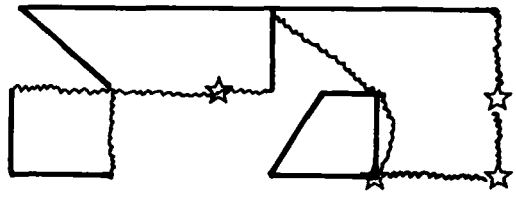
BRANCH IV

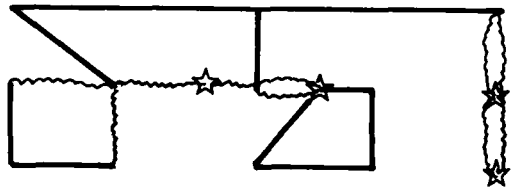
For each branch GRP we solve the mixed 1,2-matching problem subject to no subtours. The solutions are given below, where the matched arcs are denoted by wavy lines.

BRANCH I

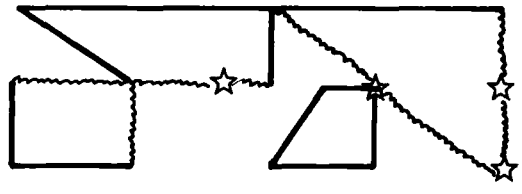


BRANCH II





BRANCH III



BRANCH IV

For each branch solution an Euler tour exists. Branch IV has the least cost Euler tour and it is the optimal TSP solution on G. We note that even if we had used the TSP subtour elimination algorithm for this problem, it is likely that subtour S_3 would have been obtained anyway.

VII. EXTENSIONS

For undirected and fully directed connected graphs we have shown that it is advantageous to solve node-oriented routing problems as equivalent GRPs. The Euler tour conditions are slightly different for directed graphs than for undirected graphs, so that is why the transportation problem is the bounding subproblem in one case, and the b-matching problem in the other case. For mixed graphs, the Euler tour conditions become considerably more complicated. In fact, just checking to see if an Euler tour exists on a mixed graph requires solving a network flow problem. The TSP can be solved on mixed graphs using the embedded assignment problem in the subtour elimination algorithm, but problems may still occur due to the large number of size 2 and 3 subtours.

The problem of finding the optimal solution to the CPP on a general mixed graph is, to the author's knowledge, still unsolved. It follows that the RPP and GRP on mixed graphs are also unsolved problems. For small problems, and for problems on mixed graphs that have predominately directed edges, all the undirected required arcs can be converted to an equivalent set of required nodes, leaving a GRP with only directed required arcs, which can be solved.

Edmonds and Johnson have given an algorithm which solves the CPP for the special class of mixed graphs that satisfy the following two conditions:

1) Every node is met by an even number of edges (directed or undirected).
2) There is no proper subset S of nodes such that every edge meeting one node in S and one node in $\bar{S}$ is directed away from the node in S.

They note that condition 2 is necessary and sufficient for a postman tour to exist. Condition 1 is necessary for the algorithm given, and it is noted that grid networks, with each node of degree four, satisfies this condition.

If condition 1 holds for the CPP on G, and every node is also symmetric (number of edges directed in equals number of edges directed out), then an Euler tour exists on G and it is the optimal CPP solution. If each node is not symmetric, then the CPP is solved by first solving a simple minimum cost network flow problem which achieves symmetry while maintaining node degree parity. The flow solution defines how the undirected arcs must be directed, and which arcs must be added to the original graph, such that an Euler tour exists on the augmented graph, which is the CPP solution. While this algorithm is important for certain applications of the CPP, condition 1 so rarely holds for the subgraph of required arcs in the RPP and GRP that this algorithm is not very useful for these problems. More work in the area of routing problems on mixed graphs is needed. There are obvious heuristics and enumeration techniques based on the exact algorithms for the directed and undirected network routing problems.

It is often the case, even for large problems, that the TSP can be converted to an equivalent GRP or RPP with a very small number of odd nodes (e.g., less than fifteen). For problems of this size, other techniques, such as dynamic programming, which would bypass the problem of subtour elimination, would be more appropriate. When the arc cost is proportional to true distance, the GRP algorithm can be easily implemented by hand on a true scale map by "eyeballing" the matching problem solution with no subtours and getting close to optimal results.

VIII. CONCLUSIONS

In this paper it has been shown how node-oriented routing and arc-oriented routing are related, and how the GRP subtour elimination algorithm is the logical extension of this relationship. This also shows the extent to which the nice properties of the Euler tour can be exploited to find tours on real networks. The reason that the GRP is more efficient than the TSP approach for real routing problems is that the TSP takes no advantage of special structure which may exist in the actual transportation network (except as it is reflected in the cost matrix).

In this paper, we have been primarily concerned with single vehicle routing problems on fully undirected connected networks. If a network is not connected, no feasible tour may exist. For connected fully directed networks, it is also possible that no feasible tour may exist. This is a severe pathology, and will cause the subtour elimination procedure to partition the solution space until it is empty.

ACKNOWLEDGEMENTS

The author would like to acknowledge the paper by Bellmore and Malone on Traveling Salesman Subtour Elimination Algorithms and the paper by Edmonds and Johnson on the Chinese Postman Problem as motivating the research described in this paper, which is part of the author's doctoral research on school bus routing and scheduling. The author would also like to thank M. Bellmore for making available his computer code for the TSP subtour elimination algorithm which included Edmonds' b-matching algorithm.

REFERENCES

1. Bellmore, M. and J. Malone, "Pathology of Traveling-Salesman Subtour-Elimination Algorithms," *Operations Research*, Vol. 19, 1971, pp. 278-307.

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

3. Berge, C., *The Theory of Graphs*, John Wiley & Sons, Inc., New York, 1962.

4. Edmonds, J., "Maximum Matching and a Polyhedron with 0,1 Vertices," *J. Res. of the National Bureau of Standards*, 696, Nos. 1-2, January-June 1965, pp. 35-40.

5. Edmonds, J. and E. L. Johnson, "Matching, Euler Tours and the Chinese Postman," *Mathematical Programming*, Vol. 5, No. 1, 1973, pp. 88-124.

6. Ford, L. R. and D. R. Fulkerson, *Flows in Networks*, Princeton University Press, New Jersey, 1962.

7. Garfinkel, R. and G. Nemhauser, *Integer Programming*, John Wiley & Sons, Inc., New York, 1972.

8. Harary, F., *Graph Theory*, Addison-Wesley, Reading, Massachusetts, 1969.

9. Hu, T. C., *Integer Programming and Network Flows*, Addison-Wesley, Reading, Massachusetts, 1969.

10. Hu, T. C., "A Decomposition Algorithm for Shortest Paths in a Network," *Operations Research*, Vol. 16, 1968, pp. 91-102.

11. Karp, R. M., "Reducibility Among Combinatorial Problems," *Tech. Report #3*, University of California, Berkeley, California, April 1972.

12. Leibman, J., "Notes on Edmonds' Matching and White's K-Matching," unpublished class notes, Johns Hopkins University, 1972.

13. Murty, K., "An Algorithm for Ranking all Assignments in Order of Increasing Cost," *Operations Research*, Vol. 16, 1968, pp. 682-687.

14. Orloff, C., "Routing and Scheduling a Fleet of Vehicles: The School Bus Problem," Ph.D. Dissertation, Cornell University, January 1973.

15. Lin, S. and B. W. Kernighan, "An Effective Heuristic Algorithm for the Traveling-Salesman Problem," *Operations Research*, Vol. 21, 1973, pp. 498-516.

Paper received November 26, 1972.