Plower Power

Chris Hartman
Kirk Hogenson
John L. Miller
University of Alaska—Fairbanks
Fairbanks, AK 99775–1110
Advisor: J.P. Lambert

Summary

We explore the problem of snow removal Wicomico County, MD, in graph-theoretic terms. We prove that Hierholzer's algorithm for generating Eulerian circuits will provide a shortest-circuit solution to the problem. We describe a computer simulation that can be modified to allow priorities to be assigned to different path choices. We generate and discuss a minimal solution and show it to be efficient.

Assumptions

Efficiency

- The cost of the snow removal is proportional to plow-truck operation time.
- The movement of residents' vehicles are hampered by unplowed roads.
- An efficient solution is one that minimizes cost and plowing time and maximizes ease of the residents' access to the roads.

Prioritization

- The number of intersections per unit area is a measure of the relative importance of a given road, since population in an area is proportional to the number of intersections in that area.
- Exits from the state highway are crucial for resident access and should be assigned a priority commensurate with an intersection-dense area.
- Dead-end roads are not as important as through roads.
- Once a road has been plowed in one direction, it is safe for limited traffic in both directions.

Plow-Trucks

- If the fuel supply carried from the garage is not sufficient, the plow-trucks may refuel during the performance of their duties without time or distance penalties.
- The speed of a plow-truck while plowing is constant. Note that since the trucks will always be plowing, the actual speed will not affect the model.
- The blade is wide enough to clear one lane of a road in a single pass and thus an entire section of road in two passes from opposite directions, since all roads are two-way and two-lane. Clearing the lane implies that the snow is pushed off the shoulder of the road.
- The plow-trucks do not break down or get stuck.
- The plow-truck operators are able to work continuously until the district is plowed, at which time relief operators may replace them.
- Traffic inconsistencies (including automobile accidents) will not impede the plow-trucks, and the effect of the plow-trucks upon traffic is not significant to the model.

Problem Analysis

Because of the similarity of a road map to a strongly connected weighted digraph, we decided to use graph theory to develop a simulation to provide the most efficient route for each of the plow-trucks.

The road map is isomorphic to a weighted digraph. A *directed graph* (or *digraph*) $G = (V, E)$ is a set V of vertices and E of edges, with each edge directed from one vertex to some other vertex and traversable in that direction only. We will assume that all graphs in this paper are finite. A *weighted digraph* is one in which a numerical weight is attached to each edge. For our model, we interpret an intersection as a vertex, *one side* of a street between intersections as an edge (so there are actually two edges for each street on the map), and the mileage between intersections as the weight for the corresponding edge.

A *trail* in a directed graph is a sequence of distinct (directed) edges for which the final vertex of one is the initial vertex of the next one. A *closed trail, or circuit,* in a directed graph is a trail in which the initial vertex of the first edge is the same as the final vertex of the last edge. We denote the edges in a circuit C by $E(C)$.

Two circuits that do not have an edge in common (though they may pass through one or more vertices in common) are said to be *disjoint*. We note that the graph of our map is *strongly connected*: for any two distinct

vertices, each vertex is reachable from the other. As long as there are no isolated vertices (as there clearly aren't in our application), this definition is equivalent to it being possible to traverse any edge by starting at any vertex and following a sequence of directed edges.

For our map, we seek an *Eulerian trail*, a closed trail that traverses every edge exactly once. An Eulerian trail that is closed is an *Eulerian circuit*.

In our problem, we would like ideally to generate routes for the trucks which are Eulerian circuits. The resulting mileage would be minimal, with no *deadheading* (retraversal of roads already traversed); and the trucks would finish at the same points from which they began. The latter feature would allow the trucks to clear the streets again, using the same route, in the case of continuous snowfall, or to return to their garages without having to traverse extra edges of the graph.

We now present the following somewhat obvious but important results, taken from Gould [1988, 126–129], with terminology and proofs adapted to the setting of directed graphs:

Lemma. *If the edges of a directed graph can be partitioned into disjoint subcircuits, then the graph has an Eulerian circuit.*

Proof: (adapted from Gould [1988, 126–127]) Suppose the edges of the graph G are partitioned into m disjoint subcircuits. We use induction to show that for any k, $(1 \leq k \leq m)$, there is a circuit in G which contains each edge of k of the subcircuits and no other edge of G. Clearly, if $k = 1$ then the statement is true. Now assume there is a circuit C that contains k of the subcircuits $(k < m)$ and no other edges of G. Since G is connected, at least one of the other subcircuits must contain a vertex of C. Let C_1 be the circuit obtained by traversing that subcircuit, beginning at some common vertex v (and thus returning there), and then following C. Then C_1 contains the edges of $k + 1$ subcircuits and no other edges. The result follows by induction. □

Theorem 1. *If each vertex of a connected directed graph has equal numbers of incoming and outgoing edges, then the graph has an Eulerian circuit.*

Proof: (adapted from Gould [1988, 126]) Denote the graph by G. We need only prove that the edges of G can be partitioned into disjoint circuits and then invoke the lemma.

If there are any edges in G, we can construct a circuit in G, starting at any edge. Because of the equality of incoming and outgoing edges, we can continue traversing edges unless we have reached our starting point.

We use induction on the number of circuits in G.

If there are no circuits, there are no edges, and the result holds trivially for the case of 0 circuits.

Now assume that the result holds for any graph with fewer than k circuits, $k \geq 1$. Suppose that G is a connected graph with all vertices having equal numbers of incoming and outgoing edges and G contains exactly k circuits.

Delete the edges of one circuit C^* of G. Each component of the remaining graph has the property that all vertices have equal numbers of incoming and outgoing edges. Further, each component has fewer than k circuits; by the induction hypothesis, their edge sets can be partitioned into disjoint circuits. The edge set of G can then be partitioned into the same circuits plus C^*. □

There are various algorithms for finding an Eulerian circuit; we use *Hierholzer's algorithm* [Gould 1988, 129]. The strategy is to produce circuits in G and patch them together properly in order to form an Eulerian circuit that completely covers G.

Hierholzer's Algorithm (adapted from Gould [1988, 129])

- *Input:* A strongly connected directed graph $G = (V, E)$, each of whose vertices has an equal number of incoming and outgoing edges.
- *Output:* An Eulerian circuit C of G.
- *Method:* Patching together of circuits.

1. Choose $v \in V$. Produce a circuit C_0 beginning with v by traversing at each step any edge not included in the circuit. (This can always be done, by the argument at the start of the proof in Theorem 1.) Set $i = 0$.
2. If $E(C_i) = E(G)$,
   then halt, since $C = C_i$ is an Eulerian circuit.
   else choose a vertex v_i on C_i which is incident to an edge not on C_i. Now build a circuit C_i^* beginning with v_i in the graph $G - E(C_i)$. (Hence, C_i^* also contains v_i.)
3. Build a circuit C_{i+1} containing the edges of C_i and C_i^* by starting at v_i, traversing C_i^* completely (hence finishing at v_i), and then traversing C_i. (Again, by the argument in Theorem 1, this can always be done.) Set $i \leftarrow i + 1$ and go to Step 2.

Theorem 2. *Hierholzer's algorithm generates an Eulerian circuit.*

Proof: (adapted from Gould [1988, 129]) The algorithm patches onto a main circuit another circuit that is by construction edge-disjoint from the main circuit. The patching hinges on the two circuits having a common vertex,

and this fact follows from the connectivity of the graph and the construction. If all the edges of the main circuit have not been used, then there must be one such edge that is incident to a vertex of the main circuit, and we use this edge to begin the next circuit to graft on. This next circuit and the main circuit thus share a common vertex.

Since each circuit patched into the main circuit increases the number of edges in the main circuit, and the graph G has only a finite number of edges, the algorithm must terminate. By the argument above, it cannot terminate without exhausting all the edges of G. At termination the algorithm clearly produces a circuit that does not contain any edges twice, by the nature of the choices of C_0 and C_i^*. Since this circuit contains all the edges of G without duplication, it is an Eulerian circuit. □

On the map associated with the snow-plowing problem, we have an equal number of incoming and outgoing edges, since we have defined an edge in such a way that there are two edges for each street: at each intersection, a street provides one incoming and one outgoing edge. Thus, we now know how to find an Eulerian circuit for the entire graph of the roads of the county—which would be just fine if we had just a single truck and wanted an efficient route for it. We now show how to benefit from having two or more trucks.

If the edges of a directed graph can be partitioned so that the induced subgraphs are each connected graphs, then Hierholzer's algorithm can be applied to each of the subgraphs separately, generating disjoint Eulerian subcircuits of the original graph. This observation allows us to extend the model to n trucks, all with the same basic algorithm. The program we give in the appendix is easily modifiable to accommodate this theoretical insight. [EDITOR'S NOTE: Omitted.]

For the two-truck case at hand, the generalized Hierholzer algorithm is the same as the original algorithm, except we have two trails to keep track of, C and D. In our snow-clearing problem, we would like the two resulting circuits to be of approximately the same length, so we add vertices to C and to D in such a way as to keep their lengths as close as possible. But when we reach Step 3 of the algorithm, it may be that we cannot find an edge connected to C which remains to be traversed — all vertices may be adjacent to D. In such a case, we would like to remove a portion of the circuit from D and add it to C, in order to allow D to absorb the unused part and still keep the distances travelled somewhat equal. In order to validate this procedure, we need the following result:

Theorem 3. *Given C and D, disjoint Eulerian subcircuits of a directed graph G, if we remove from C any circuit that is connected to D and add it to D, the modified C and D are still disjoint Eulerian subcircuits of G.*

Proof: Suppose circuit K is contained in C from vertices C_i to C_{i+j}. We

must have $C_i = C_{i+j}$, since K is a circuit. Thus we can define $C' = C - K$, a circuit. That is, C' follows C from C_0 to C_i, and then $C_i \to C_{i+j+1}$, which are adjacent, and then continues on C to $C_n = C_0$. So C' is an Eulerian subcircuit. At some point K intersects D, so we add K to D, as in Step 3 of Hierholzer's algorithm, yielding a D' that is an Eulerian subcircuit. Clearly, $E(C') \cup E(D') = E(C) \cup E(D)$. □

We now give the generalized Hierholzer algorithm for the case in which two Eulerian subcircuits are generated. The extension to n circuits is clear.

Generalized Hierholzer Algorithm

- *Input:* A strongly connected directed graph $G = (V, E)$, each of whose vertices has an equal number of incoming and outgoing edges.
- *Output:* Two disjoint Eulerian subcircuits C and D of G.
- *Method:* Patching together of circuits.

1. Choose $v_0 \in V$ and $u_0 \in V$. Produce two Eulerian circuits C_0 and D_0 as in the Hierholzer algorithm. Set $i = 0$.
2. If $E(C_i + D_i) = E(G)$,
   then halt, since C_i and D_i are two Eulerian circuits that together cover G.
   else determine which trail, C_i or D_i, is shorter; call the shorter one S, the longer one L. Choose a vertex v_i on S which is incident to an edge not on $C_i + D_i$. Now build a circuit K, beginning with v_i, in the graph of $G - E(C_i + D_i)$. If there is no such vertex v_i, go to Step 4.
3. Build a circuit S' containing the edges of S and K by starting at v_i, traversing K completely, and then traversing S. If C_i was shorter than D_i, set $C_{i+1} = S'$ and $D_{i+1} = D_i$; if D_i was the shorter, then set $D_{i+1} = S'$ and $C_{i+1} = C_i$ Set $i \leftarrow i + 1$ and go to Step 2.
4. If the circuit S does not intersect any unused edges, then we must use Theorem 3. First, add the remaining circuits to L (always possible). Then find circuits in L and add them to S until the lengths are as close as possible. Now halt, since $E(C_i + D_i) = G$.

This discussion completes the theoretical analysis of the problem. We now need to apply the generalized Hierholzer algorithm to the problem, and we would like to generate the Eulerian subcircuits in an efficient manner.

Model Design

We implemented the generalized Hierholzer algorithm, for the case of two circuits, in a C program on a DEC VAX 11/750. The program determines the most efficient route for the two plow-trucks to take, starting from the two entry points into the county. The map of the district is input as a set of edges, and both circuits are built simultaneously, minimizing variance in the total distance traveled by each plow-truck.

The simulation creates two trails, each starting from one of the asterisks on the map. The lengths of the two trails are kept approximately equal by adding edges to the shorter of the two trails at each step. When the first trail reaches either of the trail's starting points, it is not given any more edges until the other trail reaches the remaining original vertex. After both trails have come to the original starting points, the algorithm searches for circuits of unused edges. These circuits must be inserted into the trails in order to use every edge. When possible, these circuits are inserted into the shorter of the two trails.

When all of the edges are used, the program checks the lengths of both trails. If the lengths differ by more than 5 mi, the program looks for a circuit in the longer trail that can be "stolen" by the shorter trail. If such a circuit is found, it is deleted from the longer trail and added to the shorter. This process is repeated until the lengths of the trails differ by no more than 5 mi. [EDITOR'S NOTE: The authors' approach is quite satisfactory for this problem and most other practical problems. However, for some problems there may not be such relatively balanced routes. Even if there are, the authors' algorithm may not find the most balanced routes. In the worst case, their algorithm will find routes with one twice the length of the other, even though a perfectly balanced solution exists.]

During normal execution of the program, the next vertex in a given circuit is chosen according to criteria encoded in a subroutine. We expected these criteria to be the most important aspect of the program, and thought that they would have a significant effect on the efficiency of the trails. We were surprised to discover that virtually all pairs of trails which *can* be generated by the program have maximal efficiency. The restriction that the trails must be Eulerian and of approximately equal length ensures that the trails are efficient. However, the criteria can still be used to fine-tune the model. The criteria that generate the best results are, in order of importance:

1. Choose a road that has not been plowed in either direction over one that has been plowed in a single direction.
2. Choose a road that accesses a state highway over any other.
3. Choose a road that leads toward an intersection of greatest vertex degree.
4. Choose the road leading toward the vertex farthest from the *previous* vertex (not the *current* vertex).

We wrote the program in modular fashion, to allow for ease of modification of subroutines and data files. If a new road were opened in Wicomico county, the program could regenerate optimal plowing trails after the addition of only a few numbers to the data files. In addition, with minor modifications, the program could simulate the actions of more plow-trucks.

Results

We generated an optimal and efficient solution (see **Figures 1** and **2**). Truck 1 travels 122.1 miles and Truck 2 travels 125.2 miles, for a total of 247.3 miles.

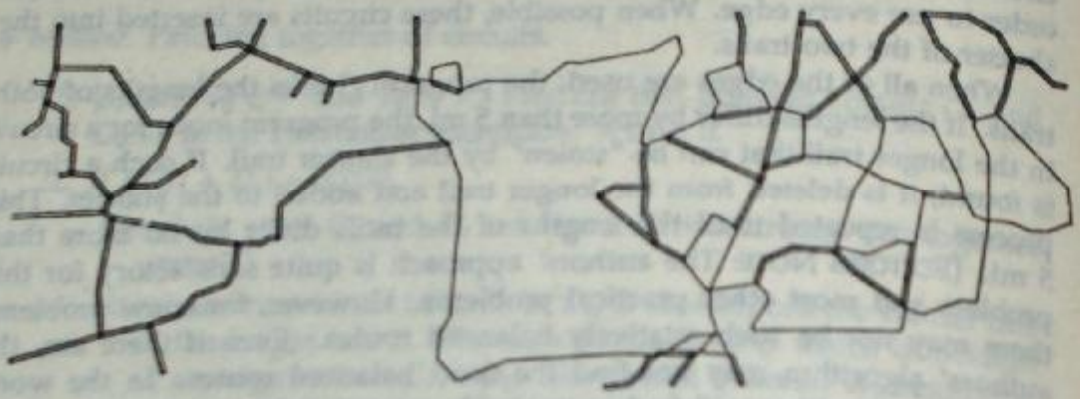

Figure 1. Truck 1's route.

Testing

We wrote computer programs to test the efficiency of the Eulerian subcircuits.

The first efficiency test finds the *number of intersections traversed* as a function of distance traveled. According to criterion (1), an efficient trail will initially traverse a larger number of intersections per unit distance than a less efficient trail.

The second efficiency test, the *weighted sector test*, divides the map into user-defined "sectors" and determines the total number of sectors that have been entered ("punctured") at least once by either trail. A trail that uniformly distributes itself across the map will enter more sectors in the initial stages of the simulation.

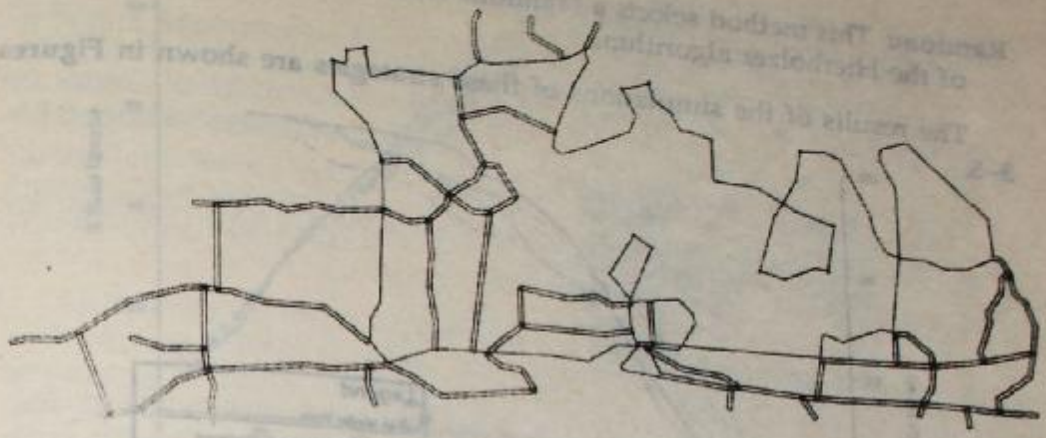

Figure 2. Truck 2's route.

The third efficiency test, which we believe to be the best test of model efficiency, is the *useable road index*. A road is considered useable if it has been plowed in one direction, and plowing again does not increase road useability in this context. Thus a trail that avoids traversing both edges of any road segment for as long as possible is more efficient under this test.

We did not develop an efficiency algorithm that evaluates the extent to which the access routes to state highways are made useable. We felt that the results to be gained from testing this feature would duplicate previous testing, i.e., there would not be much variation among the different simulations.

We chose five different strategies to subject to the efficiency tests:

Best Model Path: This is our base model; its strategy uses all four criteria for choosing the next vertex. For this strategy, we forecasted optimal results under all three efficiency tests.

Maximum Inter-node Distance: Here the only criterion used is (4), to select a vertex most distant from the previous vertex. That is, we discarded the first three criteria of the base model.

No State Highway: This is the base model without criterion (2). The trails in this simulation are not as strongly attracted to the outside bounds of the district.

Avoid Plowed Roads: This simulation combines the maximum distance criterion (4) with the stipulation of criterion (1) that singly-plowed roads are to be avoided for as long as possible.

Random: This method selects a "random" vertex, subject to the constraints of the Hierholzer algorithm.

The results of the simulations of these strategies are shown in Figures 3–5.

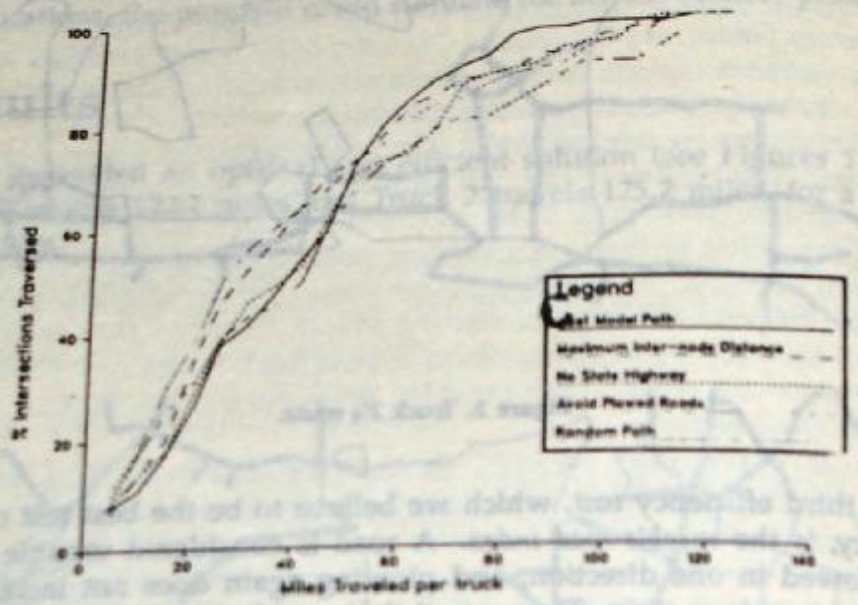

Figure 3. Intersection test.

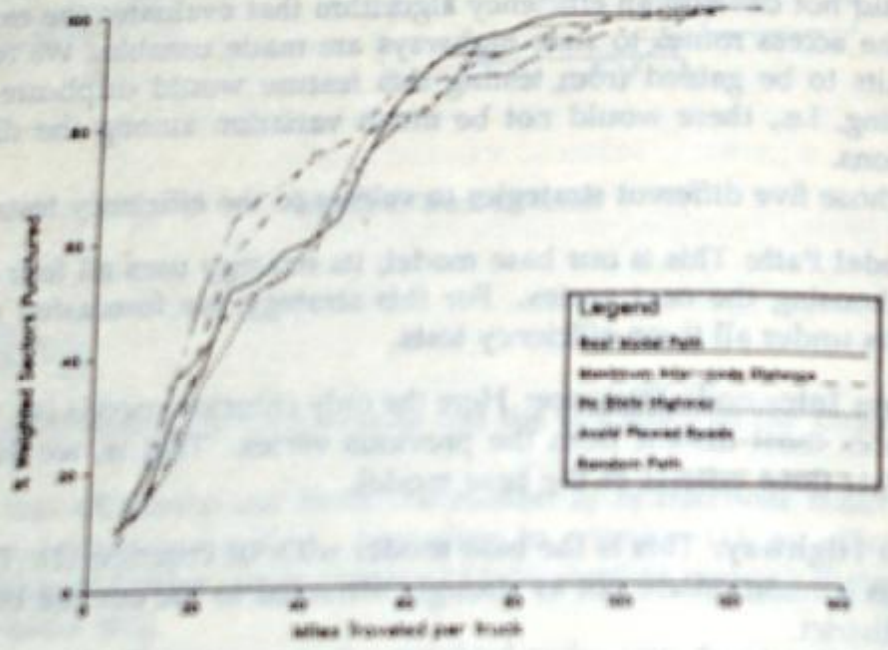

Figure 4. Weighted meter puncture test.

We note that in each figure the more-efficient simulation curves should have a higher initial slope than the less-efficient simulations.

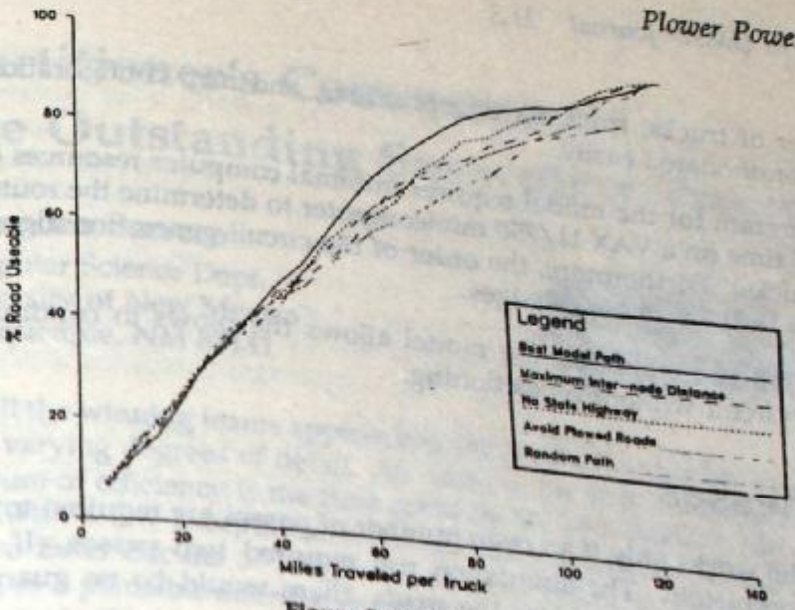

Figure 5. Useable road test.

In every case, the curves show little variation, which fact suggests that the strategy does not affect efficiency significantly. That is, what has the largest impact on the efficiency is not the efficiency criteria at all, but rather the fact that the trail is always an Eulerian circuit.

The Eulerian circuit assures us of a trail that will be optimal in what we consider to be the primary aspects: time required to plow all roads, and distance travelled by the plow-truck. The results from the efficiency tests performed on any Eulerian circuit are almost identical, regardless of how the Eulerian subcircuits C and D are chosen. Thus, the key to the model does not lie in the criteria for vertex determination; rather, it lies in the graph-theoretic basis of the simulation. When Hierholzer's algorithm is employed on an eligible graph, no matter what sort of idiotic criteria are entered, the resulting circuit is always Eulerian. By the very nature of an Eulerian circuit, we will obtain minimal distance and time values. This feature assures us of model stability.

Case in point: During program testing, we discovered that the data file of the map was incomplete, as we had not entered one of the edges. Upon remedy of this defect, the trail was completely different, yet the results under the different efficiency tests remained largely unchanged.

Strengths

- Every pair of spanning trails generated by the model will be minimal with regard to total distance.
- The model is extremely versatile, so that changes in efficiency criteria,

number of trucks, truck starting positions, and map configuration can be accommodated easily.

- The program for the model requires minimal computer resources (8 sec of CPU time on a VAX 11/750 minicomputer to determine the routes for both trucks). Furthermore, the order of the circuit-generation algorithm is linear in the number of edges.
- If snowfall is continuous, the model allows the plows to continue for another circuit without repositioning.

Weaknesses

- The model works only if an even number of passes are required to clear a road completely. The simulation run required two passes. If three passes were required to clear the roads, there would be no guarantee that an Eulerian circuit existed.
- We have no quantitative comparisons of efficiency of the Hierholzer algorithm with other algorithms for generating Eulerian circuits. The efficiency tests that we ran could not differentiate significantly between models employing different vertex selection criteria.

References

Gould, Ronald. 1988. *Graph Theory.* New York: Benjamin/Cummings.

Roberts, Fred S. 1978. *Graph Theory and Its Applications to Problems of Society.* Philadelphia, PA: SIAM.

Tucker, A.C., and L. Bodin. 1976. A model for municipal street sweeping operations. In *Modules in Applied Mathematics. Vol. 3: Discrete and System Models*, edited by William F. Lucas et al., 76–111. New York: Springer-Verlag.

Tucker, Alan. 1980. *Applied Combinatorics.* New York: Wiley.

Tucker, Walter B. 1977. A computer model of municipal snow removal. CRREL Report 77-30. Hanover, NH: United States Army Cold Regions Research and Engineering Laboratory.

_____, et. al. 1978. In *International Symposium on Snow Removal and Ice Control Research*, NRC Special Report 185, 293–302. Hanover, NH: National Research Council, Transportation Research Board.