Route Finding in Street Maps by Computers and People

R. J. Elliott

M. E. Lesk

Bell Laboratories
Murray Hill, New Jersey 07974

Abstract

We wrote a computer program which gives driving directions in northern New Jersey. Its data base combines a street map and a telephone book, so requests like "give directions from the Lackawanna Diner to the nearest dry cleaner" (properly specified) can be answered.

This problem involves both human factors and algorithmic problems. From the human factors standpoint, what kind of route is best: shortest distance, most concise directions, fewest turns, or some combination of these? And from the algorithmic standard, what is a good shortest-path algorithm: breadth-first search, depth-first search, pre-storing important routes, divide and conquer, or keep a hierarchy of maps with progressively fewer streets?

We implemented breadth-first and depth-first search both single-ended and double-ended. Double-ended search was faster in 14 of 16 examples and produced shorter or equal length routes in 13 of 16. Depth-first search was always faster than breadth-first, and produced shorter routes half the time.

We also asked eight subjects for directions on 4 maps. The 32 tries at 4 problems produced 22 different routes. People's strategies typically include finding main roads, and applying divide-and-conquer as well as depth-first search. But it is difficult to characterize the experimental subjects, since different problems caused them to try different search algorithms.

Introduction.

British studies show that 4% of all driving is wasted, done by people who are either completely lost or just not following the best route.[1] Experimentation with routefinding is both a practical and an interesting heuristic problem. We have a street map of Morris and Essex counties in New Jersey,[2] a telephone book covering parts of the area, and a program to give driving directions. A sample map of Chatham, NJ is shown in Figure 1; a map of part of Madison, NJ including some business names is shown in Figure 2. We can find routes between two places where the starting place is identified as: (a) a street number and name; (b) the intersection of two streets; or (c) the name of a business listed in our Yellow Pages with an identifiable address; and the ending place is identified in any of these ways or as (d) "nearest X" where X is the name of a Yellow Pages category. Directions can be printed, or a map showing the route drawn. The large and accurate data base eliminates the need to handle "fuzzy" positions as considered by McDermott[3] but instead forces us to worry about the implementation of the routefinder.

At first, we actually found the shortest route. However, people shown minimum-distance routes often recoiled in horror; they had far too many turns. In our area, on a trip of any length, the program would usually make turns at least every mile, having found some way to save a few yards. We introduced a criterion that each right turn cost 1/8 mile and each left turn cost 1/4 mile. With these extra penalties, minimum-distance routes look reasonable to users.

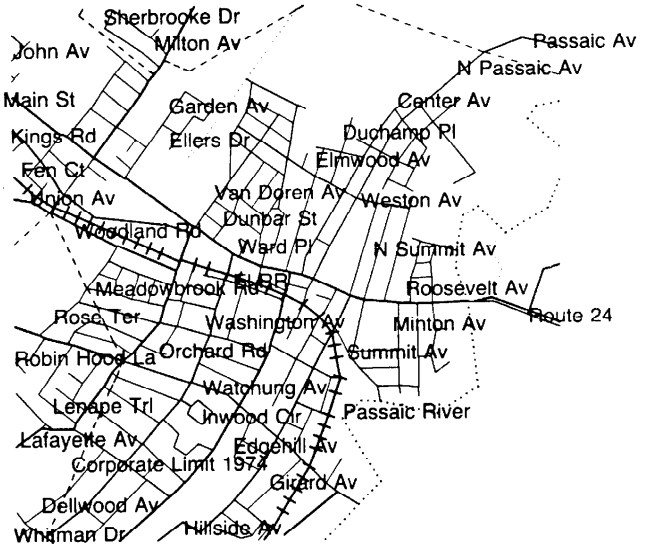

Figure 1.

Downtown Chatham, NJ

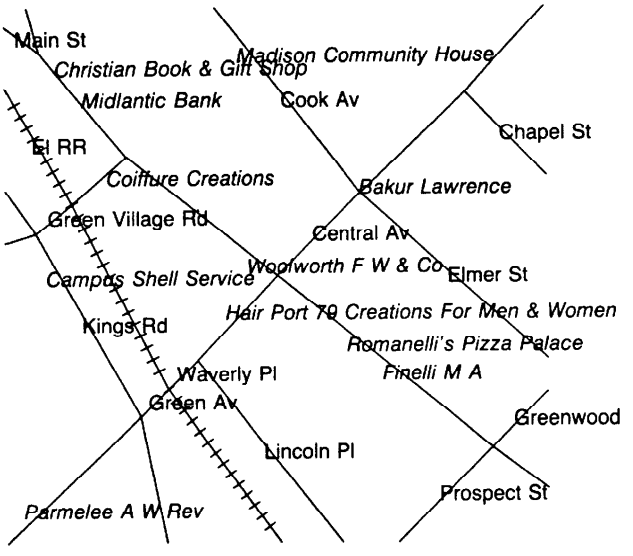

Figure 2.

Downtown Madison, NJ

Some sample routes, with drawn maps, are shown in Figures 3 and 4. In Figure 4, note that the program has gone an extra block NW on South Street in order to avoid making the double turn on Maple St, which saves less than the 3/8 mile it would be charged. In addition to handling distance minimization, the program also has facilities for handling one-way streets and limited access highways. We are currently extending the cost algorithm to include assumption of different rates of speed on different streets.

Computer algorithms.

Mathematically, this problem is familiar as the shortest-path problem.[4,5,6,7,8,9] In the substantial literature, a nondirectional breadth-first search is normally recommended. The reason is that most of the mathematicians are trying to minimize the worst-case time, and the worst case includes non-planar graphs with short edges between random points in the graph. In such a case, there is not much advantage to "directional" searching, even when directions can be defined. Furthermore, many of the graphs being used in these papers have few nodes and many edges.

Our graphs, however, are quite different. Street maps are nearly always planar (some parts of San Francisco being exceptions), and have Euclidean distances and other nice properties like triangle inequalities. More important, there are a great many nodes while edges are sparse. In our Morris and Essex street map, there are 40,000 nodes and 65,000 edges; of these 39,000 nodes and 51,000 edges involve streets (rather than rivers, town boundaries, railroads and the like). (To compare with something more familiar to people, Manhattan has 5000 nodes and 8000 edges; 4000 nodes and 7000 edges involve streets.) As a result, algorithms which touch almost every street node are too slow in our case. Even considering all the street nodes that lie in between the source and destination would prevent rapid response. (Although we retain the railroad and river information so that important landmarks are presented in directions or on drawn maps, they are not searched in route-finding).

We have compared the basic breadth-first and depth-first search algorithms, using both double-ended and single-ended searching. In breadth-first search, we do not spread out omnidirectionally; instead the search is biased in the right direction by penalizing points based on their distance from the destination. In depth-first search, we follow streets as long as the distance on this street is decreasing. The single-ended searches go from the starting point to the destination; the double-ended searches advance from both points until a meeting point in the middle is found.

Table 1 shows the rough results, averaged over ten trips in New Jersey and California.

Table 1.

Computer Routing

Method	Single-end		Double-end	
	Nodes Touched	Distance (miles)	Nodes Touched	Distance (miles)
Breadth-first	904	7.65	784	7.66
Depth-first	509	8.55	450	7.88

As can be seen, depth first search touches substantially fewer nodes (which means that it runs on average 45% faster) and yet finds, in the two-way case, almost as good routes. Double-ended searching is better than single-ended for either algorithm.

Other algorithms that might be considered: (a) Divide and conquer. It is faster to find a route from A to B and then from B to C than to find a route from A to C. Thus, a strategy of identifying points likely to be in the middle of a route would substantially simplify searching. Divide and conquer would also be effective at dealing with obstacles, such as rivers or railroads. Unfortunately the obvious algorithm for picking intermediate points (identifying street

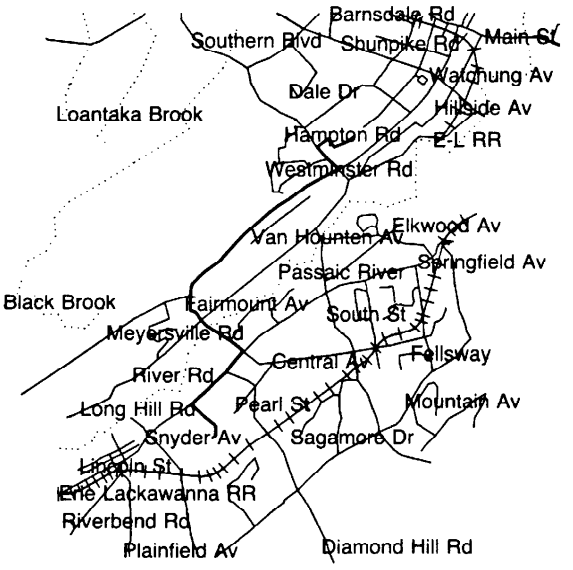

Figure 3.

Route from Lawrence Drive, Berkeley

Heights, to Hall Road, Chatham

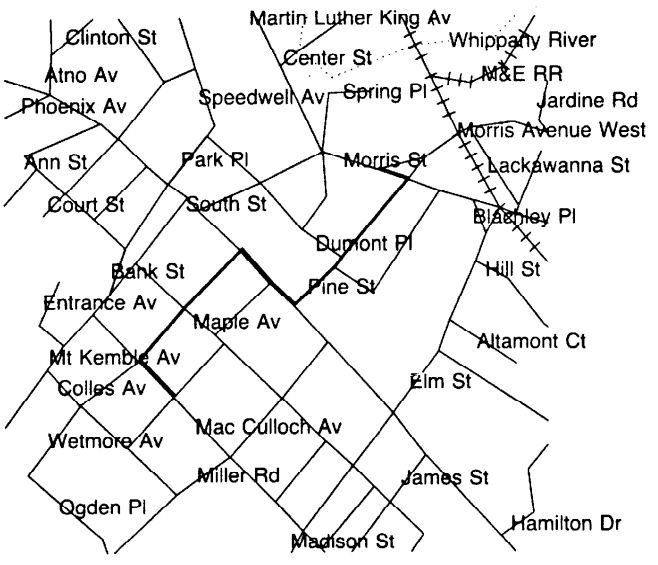

Figure 4.

Route from the Lackawana Diner to Oak St.

and MacCulloch Avenue, Morristown.

junctions roughly midway between the endpoints) is horrible on rectangular street grids; it would follow a staircase with many turns. (b) Pre-stored routes. In familiar areas, it is clear that people remember particular trips and find new routes by relating them to existing trips. Although such an "expert" system could probably be built it seems unnecessary, since people can find routes in areas they've never seen before. (c) Hierarchical search. This is probably the right answer: first go to important streets, then travel on a backbone map of important streets until you get to the destination, and then stop. As will be seen, this shares many elements with what people do, and is reasonably compatible with computer requirements.

Human algorithms.

Since the mathematical literature was not enough to solve the problem, we tried looking at what people do. Two experiments were run; in a preliminary task 15 staff members were presented with an unfamiliar routing problem; then in a more systematic protocol 8 staff members each got the same 4 problems. All problems were in unfamiliar areas (Pasadena, Dallas, San Mateo, and England). As mentioned, people varied quite widely in the routes they selected. The subjects were not specifically urged to find either the shortest or the fastest or the simplest route, but to do whatever they would do normally if asked for information. Table 2 shows some statistics on human route finding.

Table 2

Variations in Human Solutions

Problem	No. solutions (8 subjects)	Route lengths (miles)	Solution times (seconds)
Pasadena	6	4.2-6.2	30-432
Dallas	8	7.8-9.6	32-183
San Mateo	4	3.5-4.5	32-117
England	4	18.5-20.6	17-73

The total time taken by the subjects to solve all four route problems ranged from 133 seconds to 678 seconds, with a mean of 302 seconds.

In general, the longer routes represent greater use of high-speed roads, even if not going in the same direction. We note that in Dallas, where using Interstate 35 added 18% to the trip length, two people used it. In Pasadena, two people used the Foothills Freeway, with a cost of 23%. In San Mateo, the Bayshore Freeway could have been used at an added trip cost of 27%; no one did so.

The typical strategy was: first scan quickly for important roads going in the right direction. Then do depth first search, single ended, going to and along the important roads. This combines divide-and-conquer and depth-first search on the sub-problems. The first part of the procedure resembles the use of intermediate goals by Kuipers' subjects.[10] In general a "first-hit" strategy was used: as soon as one route was found people reported success. In 4 cases of the 32, though, people tried again and found a second route. People seem to have some sense of how long a trip "ought" to take, and they search longer, back-up, or use more breadth-first searching when they think they have found an unacceptable result. Two people did one double-ended search. Breadth-first search was used in complex cases to find the way to the expressway. Typically, about half the time used in finding a route was spent looking for major roads and otherwise in "planning;" the other half was spent tracing out the route.

The printing on the map affects the route. People depend on color and presentation, not labels, to find main streets. The English map was condemned as unfamiliar (and one person tried at first to go along a railway). The American street maps (all Rand McNally or Gousha) were criticized as not indicating expressway entrances clearly enough. People always began searching at the starting point. Thus, if the starting point was on a main road, people would always go along it first. A destination near a main road was less likely to cause people to use that road.

On the Pasadena trip, the average human took 98 seconds to find a route 5.4 miles long, requiring 3.3 turns (even excluding one particularly slow human, the remaining seven subjects took an average of 56 seconds). The computer, using two-ended depth-first search, took 12 seconds of VAX 11/750 CPU time to find a route 4.2 miles long, requiring 3 turns. Figures 5 and 6 show some human and computer routes, respectively. Thus, in this example (representing people unfamiliar with an area), using computer routing produces a saving of 24% in gasoline and 10% in the chance of getting lost. However, we should note that a human expert (someone who went to graduate school in Pasadena) took 24 seconds to find a route of 4.5 miles involving 3 turns; and a human grandmaster (born and brought up in Pasadena) found a similar trip without even looking at a map.

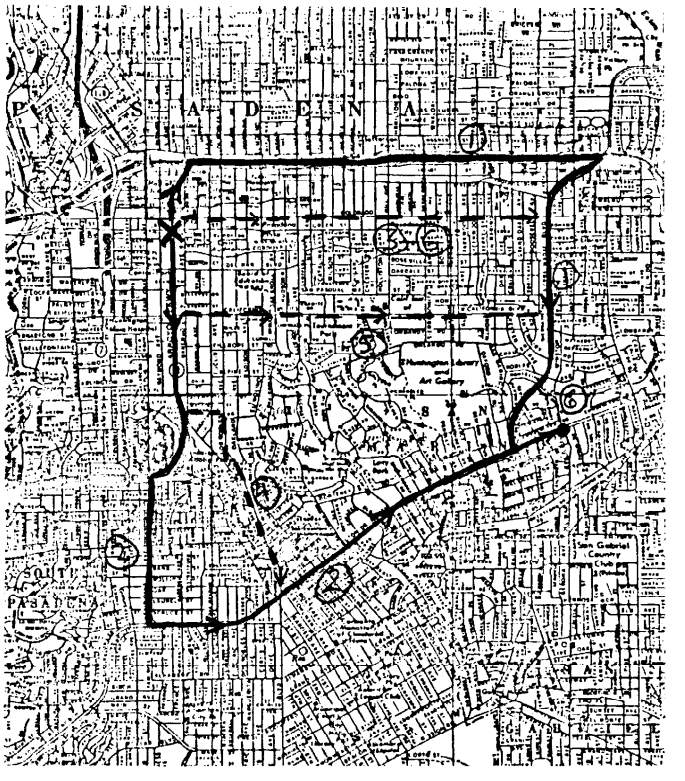

Figure 5.

Human routes in Pasadena.

Routes 1 and 2 were each followed by two subjects; routes 3 through 6 were each chosen by one subject.

Conclusions.

On one level, we have a route-finding program. Depth-first search seems to be most effective. Work continues on smoothing out the interface so a greater variety of requests can be handled (e.g. giving synonyms for category names) and speeding up the routing. But on another level, note that in this practical problem, an almost exact mathematical analogy turns out to have mostly unproductive literature solutions. Since these solutions are valuable in other kinds of shortest path problems, it emphasizes the importance of the problem domain in picking algorithms.

References

1. *Computerworld*, p. 7 (Dec. 22, 1980).
2. *Geographic Base File GBF/DIME, 1980: Technical Documentation*, U. S. Department of Commerce, Data Users Services Division, Washington, D. C. (1980).
3. D. McDermott, "A Theory of Metric Spatial Inference," *Proc. First National AI Conference*, Stanford, California, pp. 246-248 (1980).
4. R. W. Floyd, "Algorithm 97: Shortest Path," *Comm. Assoc. Comp. Mach.* **5**, p. 345 (1962).
5. A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Mass. (1974). pages 207-209
6. D. B. Johnson, "Efficient Algorithms for Shortest Paths in Sparse Networks," *J. Assoc. Comp. Mach.* **24**(1), pp. 1-13 (1977).
7. C. Witzgall, J. F. Gilsinn, and D. R. Shier, "Shortest paths in networks," pp. 171-255 in *Case Studies in Mathematical Modeling*, ed. W. E. Boyce, Pitman, Boston (1981).
8. S. E. Dreyfus, "An Appraisal of Some Shortest-Path Algorithms," *Operations Research* **17**(3), pp. 395-412 (May-June 1969).
9. R. E. Tarjan, "Fast Algorithms for Solving Path Problems," *J. Assoc. Comp. Mach.* **28**(3), pp. 594-614 (1981).
10. B. Kuipers, "Modeling Spatial Knowledge," *Cog. Sci.* **2**(2), pp. 129-153 (1978).

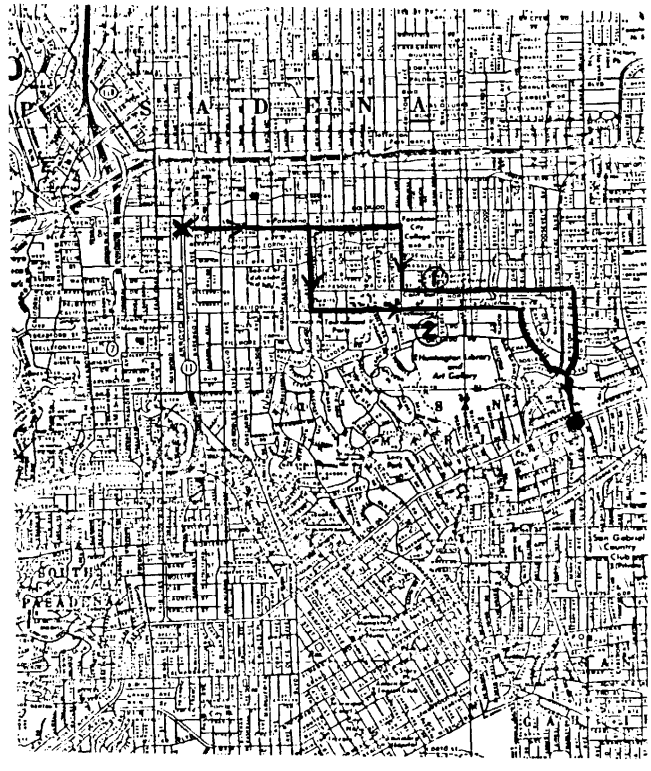

Figure 6.

Computer routes in Pasadena.

Route 1: Depth-first search.
Route 2: Breadth-first search (3.9 miles).