Finding Alternatives to the Best Path

Kelley Scott and David Bernstein
Princeton University

New Jersey TIDE Center
Directed by Prof. Louis J. Pignataro
New Jersey Institute of Technology
Newark, NJ
pignataro@admin.njit.edu

A great number of algorithms have been developed for finding the "best" path through a network, where "best" can be defined in terms of time, cost, distance, or some combination of the three [see (1) for a recent review and evaluation]. Not surprisingly, many of these algorithms have been and are now being used in Advanced Traveler Information Systems (ATIS) and Advanced Traffic Management Systems (ATMS) [see, for example, (2) and (3)].

While these algorithms have proven to be quite useful in ATIS and ATMS, they are often not sufficient. In particular, there are many situations in which it is necessary to generate alternatives to the best path. In pre-trip planning, for example, drivers often want to be provided with one path for their outbound trip and an alternative path for their return. As another example, users of in-vehicle guidance systems often want to be provided with several alternative paths that they can use to avoid particular facilities (e.g., because of an incident on that facility). As a final example, traffic control systems often need to spread the traffic between two points over multiple paths in order to reduce congestion.

Hence, an important task in this project has been the development of algorithms for finding alternatives to the best path between a given origin and destination. In this paper we discuss a constrained shortest path problem that can be used to generate alternative paths. We also discuss an efficient algorithm for generating such paths and present initial computational studies.

EXISTING APPROACHES

Two approaches for finding alternative paths are commonly proposed: the first involves link elimination, and the second involves finding the r-best paths.

In link elimination, the traveler or operator is first provided with the best path. Then, if an alternative is needed, the traveler/operator is asked which links in the best path should be excluded from the alternative. For example, consider the network shown in Figure 1 where the numbers next to each link indicates the travel time on that link. The fastest path from O to D is shown in dark grey.

Suppose that the user wanted an alternative path that did not include the last link in the best path. Then, this link could be "eliminated" from the network (or temporarily given a very large travel time) and the new best path could be found. This path is illustrated in light grey in Figure 2. The advantage of this approach is its efficiency—the alternatives can be generated as easily as the best path. The disadvantage is that, in many instances, the traveler/operator may not be "unhappy" with particular links but with the path as a whole (in some difficult

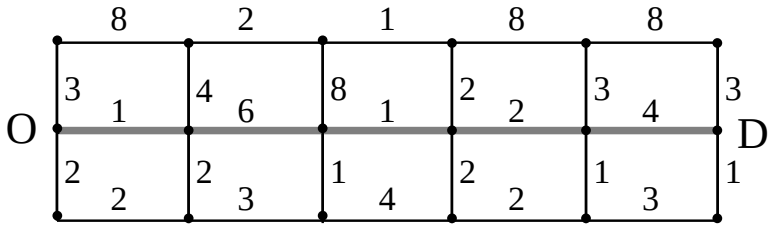

Figure 1: The Best Path from O to D

to define way).

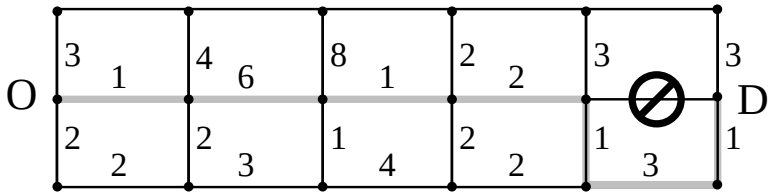

Figure 2: The Best Path from O to D with a Link Eliminated

The other approach that has been frequently proposed is to find the first r-best paths and present the traveler/operator with some or all of them. While this method makes sense in principle, in practice it has two major flaws. First, r-best path algorithms tend to be fairly slow. For example, finding the seventh-best path is often considerably more difficult than finding the best path. Second, the paths identified by such algorithms tend to be very similar. This can easily be illustrated using the previous example. In Figure 3, the best path from O to D is highlighted in dark grey and the second fastest path is highlighted in light grey. These two paths are almost identical and, as a result, travelers/operators may not view them as true alternatives.

A NEW APPROACH

As part of this effort we have been trying to develop new methods for generating alternative paths that have all of the advantages of the two existing approaches but none of the disadvantages. The specific method we will discuss here is based on the idea that when travelers/operators ask for alternative paths they want paths

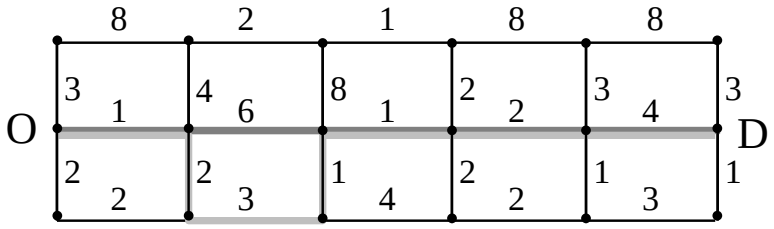

Figure 3: The Best and Second-Best Paths from O to D

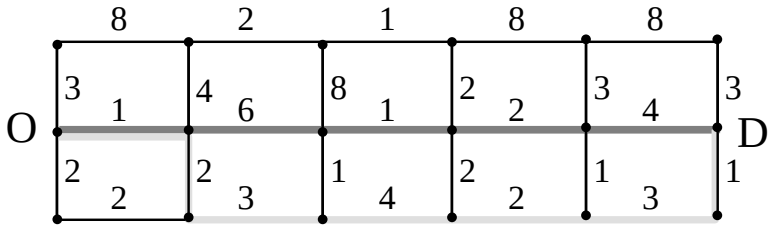

Figure 4: The Best Path and the Corresponding Best 1-Similar Path

that do not have "many" links in common but they do not want to identify the links that should be different. Therefore, we have taken the traditional best path problem and added a constraint that prevents the solution from being too similar to the true best path.

To make this idea somewhat more formal, we say that a path p is k-similar to a path s if p and s have at most k links in common. Now, if z denotes the best path, what we want to find is the best k-similar path to z.

Returning to the above example, Figure 4 shows the best path in dark grey and the corresponding best 1-similar path in light grey (i.e., the "next best" path that has at most 1 link in common with the "best" path).

As it turns out, this problem is very easy to define mathematically. Consider a *network*, $\mathcal{G}$, comprised of a finite set of *nodes*, $\mathcal{N} = \{1, \ldots, m\}$, and a finite set of (directed) *links*, $\mathcal{L} = \{1, \ldots, n\}$. The *node-arc incidence matrix* for $\mathcal{G}$, which is denoted by A, has components a_{ij} defined as follows: $a_{ij} = 1$ if link j is directed out of node i, $a_{ij} = -1$ if link j is directed into node i, and $a_{ij} = 0$ otherwise.

On this network there is a single origin node, O, and a single destination node, D. We use the vector b to indicate the origin and destination. Specifically, we let $b_O = 1$, $b_D = -1$, and $b_i = 0, i \in \mathcal{N} - \{O, D\}$. Thus, any x that satisfies:

$$\begin{array}{rcl} Ax & = & b \\ x & \in & \{0,1\}^n \end{array} \tag{1}$$

corresponds to a *path* from O to D.

We assume that the (abstract) *cost* on all links is known and given by $c = (c_j : j = 1, \ldots, n)$. We further assume that the cost on a path corresponding to x is given by:

$$C(x) = c^\top x. \tag{2}$$

Given this, it is well-known that the *minimum cost path problem* can be formulated as:

$$\begin{array}{rcl} \min_x & c^\top x & \\ \text{s.t.} & Ax & = & b \\ & x & \geq & 0 \end{array} \tag{3}$$

where the constraint $x \geq 0$ can replace the constraint $x \in \{0,1\}^n$ because this problem has the *integrality property* (i.e., because A is *totally unimodular*).

Now, suppose z is a solution to (3). Then, given another path x, it is easy to see that

$$z^\top x = \sum_{i=1}^{n} z_i x_i \tag{4}$$

is the number of links that z and x have in common. Hence, the problem of finding a minimum cost path that is k-similar to z is given by:

$$\begin{array}{rcl} \min_x & c^\top x & \\ \text{s.t.} & Ax & = & b \\ & z^\top x & \leq & k \\ & x & \in & \{0,1\}^n \end{array} \tag{5}$$

That is, the solution to this problem will be a minimum cost path that has at most k links in common with path z.

FINDING THE BEST K-SIMILAR PATH

Unlike the traditional minimum cost path problem given in (3), problem (5) must explicitly include the constraints $x \in \{0,1\}^n$ since it no longer has the integrality

property. (If we were only to impose the constraint $x \geq 0$ we could get non-integral solutions, which are meaningless in the context of x representing a path.) The introduction of the link overlap constraint thus makes finding the best path much more difficult [i.e., shortest path problems with a single constraint are NP-hard (4)].

We can, however, "solve" this problem using Lagrangian Relaxation [see (5), (6) and (7), along with (4)]. First, we bring the overlap constraint into the objective function with a scalar multiplier, λ, as follows:

$$\begin{array}{rrcl} \min_x & c^\top x + \lambda(z^\top x - k) & & \\ \text{s.t.} & Ax & = & b \\ & x & \geq & 0 \end{array} \tag{6}$$

This additional term in the objective function serves to penalize violations of the constraint. Observe that, for any particular λ, the solution to (6) serves as a lower bound to (5). Since we would like to find the greatest possible lower bound (i.e., the solution itself), the problem we wish to solve is:

$$\begin{array}{rrrcl} \max_\lambda & \min_x & c^\top x + \lambda(z^\top x - k) & & \\ & \text{s.t.} & Ax & = & b \\ & & x & \geq & 0 \end{array} \tag{7}$$

The solution to this maximization problem is not guaranteed to be the optimal solution of the original problem but, as a heuristic, the relaxation technique can work quite well. The problem given in (7) is concave in λ, hence it is relatively easy to solve. The only potential difficulty is that the objective function is not differentiable in λ. Fortunately, in this case λ is a scalar, and we can use any one-dimensional search algorithm that does not require derivatives (e.g., Dichotomous Search, Fibonacci Search, Golden Section Method).

Whichever search algorithm is used, problem (6) must be solved for each "test value" of λ. One could, of course, solve it using any linear programming solver since relaxing the overlap constraint restores the integrality property. However, this problem can more easily be solved using a best path algorithm. Specifically, letting $\tilde{c} = c + \lambda z$, one need only solve a minimum cost path problem using $\tilde{c}$ rather than c. In this modified best path problem, the links comprising the best path are penalized by λ—as λ increases, the number of links in common with the best path decreases. Depending on the algorithm used, it may even be possible to "warm start" using information obtained when solving for z.

In addition, we must specify an initial search interval for λ. Observe that,

for some $\lambda' > 0$ and two distinct paths x_1 and x_2, $\tilde{c}^\top x_1 = (c + \lambda' z)^\top x_1 = (c + \lambda' z)^\top x_2 = \tilde{c}^\top x_2$. Alternatively,

$$\lambda' = \frac{c^\top x_2 - c^\top x_1}{z^\top x_1 - z^\top x_2}$$

We can therefore obtain an upper bound for λ' by computing the difference in costs on the best path and on a path having no links in common with the best path, then dividing by the smallest difference in the number of links shared with the best path (i.e., 1). We find the disjoint path by temporarily setting the cost on the links in the best path to "infinity" and solving once again for the shortest path. The length of the uncertainty interval used to terminate the search was set at 0.0001 in all cases.

NUMERICAL RESULTS

To provide evidence of the efficiency of this approach, we implemented the heuristic in C and examined its performance on several test cases. For these computational studies, we used the Golden Section Method for the one-dimensional optimization, and used a binary-heap implementation of Dijkstra's algorithm [(8), (9), (10)] to solve the shortest path subproblem.

The test network, shown in Figure 5, is the New Jersey highway system; it is comprised of 321 nodes and 1124 arcs. The cost associated with each arc is simply the travel time (measured in hours). Table 1 lists the categories of roads and the travel speed assumed for each type.

	Number	Speed (mph)
Ordinary road	666	30
Divided highway	190	40
Freeway	158	50
Toll road	104	50
Toll bridge or tunnel	6	25

Table 1: New Jersey Road Types

Fourteen representative "trips" on the network with varying origins, destinations, and number of links in the minimum time path were selected for study. They are summarized in Table 2.

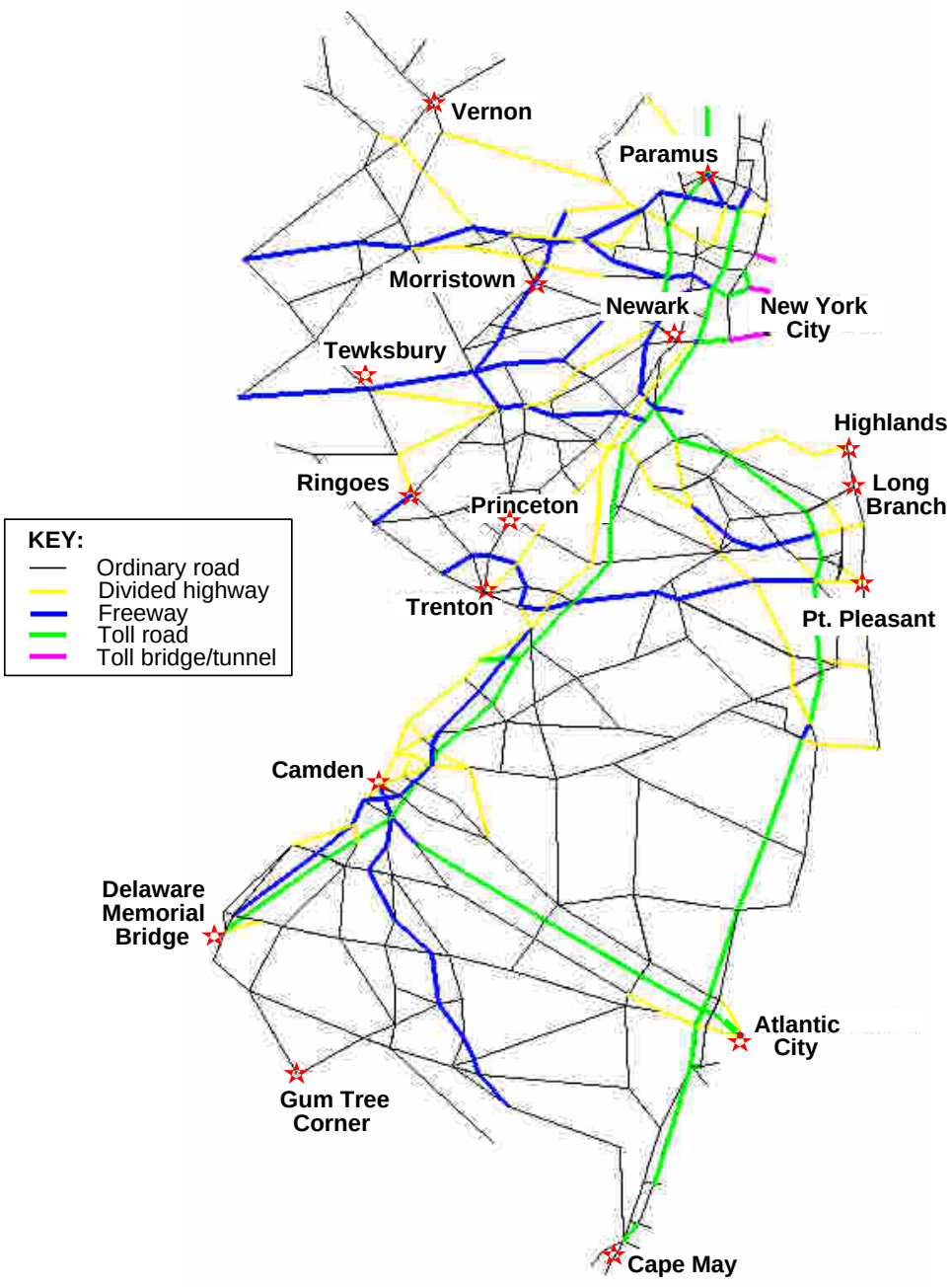

Figure 5: Test Network

Pair	Origin	Destination	Best Cost (hrs)	Links in Best Path
1	Paramus	Atlantic City	3.610	25
2	Princeton	Cape May	3.836	24
3	Tewksbury	Atlantic City	3.783	21
4	Camden	New York City	2.394	16
5	Delaware Memorial Bridge	New York City	3.029	14
6	Princeton	Vernon	2.408	14
7	Trenton	Highlands	1.967	13
8	Princeton	New York City	1.442	12
9	Newark	Pt. Pleasant	1.668	11
10	Newark	Ringoes	1.679	10
11	Gum Tree Corner	Camden	1.773	9
12	Newark	Long Branch	1.560	9
13	Gum Tree Corner	Atlantic City	2.464	8
14	Princeton	Morristown	1.204	7

Table 2: Examples

We imposed the same set of four overlap constraints on each trip and solved the corresponding constrained shortest path problems using both our relaxation method and an implementation of Yen's algorithm for the r-shortest path problem (11). For each origin-destination (OD) pair and limit on shared links, Tables 3 and 4 list the cost on the best path and the results of the heuristic and of the r-best algorithm. We halted the latter method after 2000 shortest path calls had been made.

The relaxation method appears to be very efficient, requiring only a handful of shortest path calls and, using a Silicon Graphics Indy workstation with a MIPS R4600 CPU running at 100 MHz and 32 Mb of memory, less than 0.2 seconds of CPU time to solve every problem. In several of the 56 trials, the heuristic does fail to provide the true minimum cost path that meets the overlap constraint; the path costs in these instances are italicized. However, in all of these cases, the duality gap was always less than 10 percent of the cost on the minimum time path.

Note that the number of shortest path calls is *independent of the overlap constraint* and, to a large degree, of the length of the best path; instead, the difference in costs on the best and disjoint paths relative to the minimum cost is the primary factor influencing the efficiency. For example, for OD pair 14, the number of

OD Pair	k	Best k-Similar Path Cost	Relaxation Heuristic Cost on Solution	Shared Links	No. of SP Calls	r-Best Algorithm Cost on Solution	r	No. of SP Calls
1	6	3.953	3.953	4	25	-	-	>2000
	3-1	4.212	4.212	0	″	-	-	″
2	6	3.912	3.912	6	24	3.912	9	171
	3	4.027	4.027	3	″	4.027	61	1359
	2	4.126	*4.208*	1	″	-	-	>2000
	1	4.208	4.208	1	″	-	-	″
3	6-2	3.817	3.817	2	20	3.817	3	42
	1	3.843	3.843	0	″	3.843	5	85
4	6	2.557	2.557	6	24	2.557	41	708
	3	2.722	*2.764*	2	″	-	-	>2000
	2	2.722	2.722	2	″	-	-	″
	1	2.819	2.819	1	″	-	-	″
5	6	3.251	*3.358*	3	26	3.251	71	1300
	3	3.358	*3.413*	2	″	-	-	>2000
	2	3.413	3.413	2	″	-	-	″
	1	3.839	3.839	1	″	-	-	″
6	6	2.612	*2.620*	4	25	2.612	30	426
	3	2.716	2.716	2	″	2.716	107	1565
	2	2.716	2.716	2	″	2.716	107	1565
	1	2.861	2.861	1	″	-	-	>2000
7	6-1	1.990	1.990	0	18	1.990	2	**14**
8	6	1.639	*1.643*	6	21	1.639	20	237
	3	1.747	*1.791*	3	″	1.747	84	1100
	2	1.791	1.791	2	″	1.791	144	693
	1	1.846	1.846	1	″	1.846	-	>2000
9	6-3	1.820	1.820	3	24	1.820	25	302
	2	1.878	1.878	2	″	1.878	54	693
	1	1.962	*2.046*	1	″	1.962	163	>2000

Table 3: Performance Comparison

OD Pair	k	Best k-Similar Path Cost	Relaxation Heuristic: Cost on Solution	Shared Links	No. of SP Calls	r-Best Algorithm: Cost on Solution	r	No. of SP Calls
10	6-3	1.714	1.714	3	21	1.714	2	**11**
	2	1.734	1.734	2	″	1.734	4	30
	1	1.764	*1.784*	0	″	1.764	7	60
11	6	1.791	1.791	6	20	1.791	2	**10**
	3-1	1.834	1.834	0	″	1.834	3	**19**
12	6	1.581	1.581	6	25	1.581	4	30
	3	1.736	1.736	3	″	1.736	21	210
	2	1.821	*1.949*	0	″	1.821	51	545
	1	1.932	*1.949*	0	″	1.932	112	1292
13	6	2.516	2.516	5	25	2.516	5	28
	3	2.569	*2.574*	2	″	2.569	10	88
	2	2.574	2.574	2	″	2.574	12	111
	1	2.612	2.612	1	″	2.612	18	174
14	6	1.299	1.299	5	25	1.299	2	**8**
	3	1.552	*1.633*	2	″	1.552	19	152
	2	1.633	1.633	2	″	1.633	29	252
	1	1.764	1.764	1	″	1.764	86	834

Table 4: Performance Comparison, cont'd

shortest path calls was greater than that required for OD pair 7, although the latter had roughly twice as many links in its best path.

However, the number of shortest path calls required by the r-best method does depend on the value of the overlap constraint. Specifically, if the constraint k is comparable to the number of links in the best path, the r-best method tends to solve the problem relatively quickly (the number of shortest path calls are shown in bold in these cases). For OD pairs with more links in the best path, the r-best method usually must generate a much greater number of paths before finding one that meets the same constraint. As the constraint is tightened, the r-best method tends to do even more poorly. Since the number of shortest path calls required by the heuristic is the same regardless of the "tightness" of the overlap constraint, the relaxation method typically performs much better on these problems.

Exceptions exist, of course. In the trials for OD pair 7, the second-best (and third-best, etc.) path was disjoint. As a result, the relaxation method and r-best

path method yielded the same solutions for every constraint, with the latter method requiring slightly fewer calculations. In addition, the r-best method "won" all 4 trials associated with OD pair 11. In a quarter of the trials, however, over 2000 shortest path calls were made without the r-best method providing a solution at all. For OD pair 1, for example, even the least stringent constraint (i.e., 6 shared links) could not be met within this limit.

In general, for OD pairs for which the constraint is small with respect to the number of links, the relaxation technique will be the faster option. Furthermore, the heuristic stores at most 3 paths and is thus less memory-intensive.

CONCLUSIONS

Advanced Traveler Information Systems and Advanced Traffic Management Systems often require non-traditional path generation/finding methods. This paper has considered one example of this – the generation of alternatives to the best path. We have shown that existing methods of identifying alternative paths (i.e., link elimination methods and r-best methods) can be less than ideal, and we have proposed an alternative based on the notion of k-similar paths. We have also shown how Lagrangian Relaxation methods can be used to efficiently identify optimal or near-optimal solutions.

REFERENCES

1. Cherkassky, B.V., A.V. Goldberg, and T. Radzik. "Shortest Path Algorithms: Theory and Experimental Evaluation," *Mathematical Programming*, Vol. 73, pp. 129-174, 1996.

2. Ziliaskopoulos, A. and H.S. Mahmassani. "Time-Dependent, Shortest-Path Algorithm for Real-Time Intelligent Vehicle Highway System Applications," *Transportation Research Record*, Vol. 1408, pp. 94-100, 1993.

3. Kaufman, D.E. and R.L. Smith. "Fastest Paths in Time-Dependent Networks for IVHS Application," *IVHS Journal*, Vol. 1, pp. 1-12, 1993.

4. Handler, G.Y, and I. Zang. "A Dual Algorithm for the Constrained Shortest Path Problem," *Networks*, Vol. 10, No. 4, pp. 293-310, 1980.

5. Held, M. and R. Karp. "The Traveling Salesman Problem and Minimum Spanning Trees," *Operations Research*, Vol. 18, pp. 1138-1162, 1970.

6. Held, M. and R. Karp. "The Traveling Salesman Problem and Minimum Spanning Trees: Part II," *Mathematical Programming*, Vol. 6, pp. 62-88, 1971.

7. Geoffrion, A. "Lagrangian Relaxation for Integer Programming," *Mathematical Programming Study*, Vol. 2, pp. 82-114, 1974.

8. Dijkstra, E.W. "A Note on Two Problems in Connection with Graphs," *Numerische Math.*, Vol. 1, pp. 269-271, 1959.

9. Dial, R. "Algorithm 360: Shortest Path Forest with Topological Ordering," *Communications of ACM*, Vol. 12, pp. 632-633, 1969.

10. Tarjan, R.E. *Data Structures and Network Algorithms*, CBMS 44 (Pennsylvania: SIAM, 1983).

11. Yen, J.Y. "Finding the K Shortest Loopless Paths in a Network," *Management Science*, Vol. 17, No. 11, pp. 712-716, 1971.