Solving a Best Path Problem when the Value of Time Function is Nonlinear

Kelley Scott and David Bernstein
Princeton University

November 1997

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

INTRODUCTION

The shortest path problem arises in a variety of disciplines and hence has received a great deal of attention in the literature. [For a recent review and analysis, see (1) and (2).] In most of this work, however, little attention has been given to the particular measure of "distance" that is used. Indeed, the terms "distance", "cost", and "time" are often used interchangeably.

Incorporating both time and cost need not complicate the problem at all – when the value of time is fixed, for example, the "length" of a path is simply a linear combination of travel time and direct costs incurred on the path. However, in many transportation applications, it is unreasonable to assume that people have a fixed value of time. Indeed, most drivers place almost no value on saving small amounts of time but attach much more importance to greater amounts of time (3). Though at first glance this seemingly trivial observation might not appear to be problematic, it serves to invalidate most existing algorithms for solving the shortest path problem. Specifically, it results in violations of Bellman's Principle of Optimality (4).

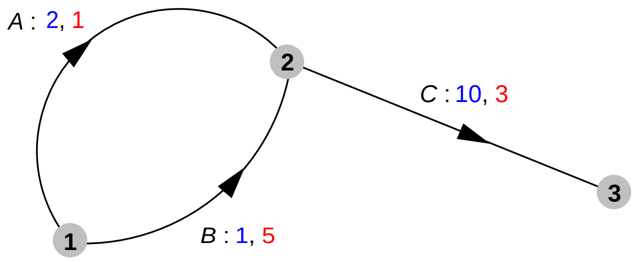

Figure 1: Simple Network with Nonadditive Path Costs

To illustrate the impact of a nonlinear cost function on the shortest path problem, consider the 3-node, 3-link network shown in Figure 1, where the first number next to each link represents the time and the second indicates the toll. Now, suppose that the aggregate cost on a path is given by the time squared plus tolls. Observe that the cheapest path between nodes 1 and 3 uses arcs B and C. (The total cost on this path is \$129 vs \$148 on the path that uses arcs A and C.) Under Bellman's Principle, one would expect that the cheapest path from node 1 to 2 also uses arc B, but this isn't the case. Since the cost on arc A is \$5 and the cost on arc B is \$6, the minimum cost path from node 1 to node 2 is actually arc A.

Denied the ability to use labelling algorithms to directly solve the nonadditive shortest path problem, one obvious recourse is to use "brute force." This entails finding the minimum time path and checking to see if it includes any tolled links. If so, a tolled link can be removed from consideration, and the shortest path recalculated. If tolled links are again included in the next solution, the process can be repeated until a "toll-free"minimum time path has been found or the inclusion and exclusion of all tolled links in the network have been considered. Obviously, if the number of tolled links is denoted by T, this method can generate as many as 2^T paths, some of which include tolled links that have already been considered in prior iterations. A slightly more elegant version would keep track of which tolled links have been found so far, and would remove combinations of these arcs from further consideration. However, this is computationally burdensome as well. A third approach to solving this problem is to formulate it as a integer program and employ branch and bound techniques.

In this paper, we show that a relatively simple and efficient option exists. Though this method still has non-polynomial worst-case complexity, in practice it appears to find the optimal solution to the nonadditive minimum cost path problem in a small number of iterations. The approach consists of solving a succession of shortest path problems, each having a single toll constraint inherited from the previous problem. Section 2 contains a formulation of the problem and a description of this solution approach, which we call the Inherited Constraint Algorithm, and the third section presents a technique for improving the algorithm's performance. Section 4 discusses an efficient method for solving the subproblems, and the fifth section provides initial numerical results. Conclusions and directions for further research are presented in Section 6.

INHERITED CONSTRAINT ALGORITHM

Consider a *network*, $\mathcal{G}$, comprised of a finite set of *nodes*, $\mathcal{N} = \{1, \ldots, m\}$, and a finite set of (directed) *arcs* (or links), $\mathcal{A} = \{1, \ldots, n\}$. 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 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.

In the traditional formulation of the problem one assumes that the *cost* on all arcs is known and given by $c = (c_j : j = 1, \dots, n)$, and that path costs are additive. Thus, the cost on a path corresponding to x is given by:

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

and the *minimum cost path problem* can be formulated as:

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

where the constraint $x \in \{0, 1\}^n$ can be replaced by the constraint $x \geq 0$ because of the total unimodularity of A.

In this paper, we assume that the cost on a path corresponding to x is given by:

$$C(x) = v\left(t^\top x\right) + \tau^\top x \tag{4}$$

where $t \in R^n_+$ and $\tau \in R^n_+$ are the vectors of *times* and *tolls*, respectively, on the arcs, and $v : R_+ \to R_+$ denotes the *value of time* function.

The problem we want to solve is thus:

$$\begin{array}{rrcl} \min_x & v(t^\top x) + \tau^\top x & & \\ \text{s.t.} & Ax & = & b \\ & x & \in & \{0, 1\}^n. \end{array} \tag{5}$$

Obviously, when $v(x) = t^\top x$ this is a simple minimum cost path problem with a composite cost vector $c = t + \tau$. We want to consider the case when v is nonlinear.

For the moment, let us act as if we have a solution to (5), which we denote by x^*. It is clear that x^* is a solution of (5) if and only if it is also a solution of:

$$\begin{array}{rrcl} \min_x & v\left(t^\top x\right) & & \\ \text{s.t.} & Ax & = & b \\ & \tau^\top x & = & \tau^\top x^* \\ & x & \in & \{0, 1\}^n. \end{array} \tag{6}$$

This further implies that x^* is a solution of (5) if and only if it is also a solution of:

$$\begin{array}{rrcl} \min_x & t^\top x & & \\ \text{s.t.} & Ax & = & b \\ & \tau^\top x & = & \tau^\top x^* \\ & x & \in & \{0, 1\}^n \end{array} \tag{7}$$

since v is strictly increasing.

Of course, we do not have a solution to (5) and if we did we would not be interested in (7). It should be intuitively clear, however, that educated guesses for x^* might allow us to move towards an optimal solution.

To see how, we let Z denote the value of $\tau^\top x^*$, the toll on the optimal path. Hence our problem becomes:

$$\begin{array}{rrcl} \min_x & t^\top x & & \\ \text{s.t.} & Ax & = & b \\ & \tau^\top x & = & Z \\ & x & \in & \{0,1\}^n. \end{array} \tag{8}$$

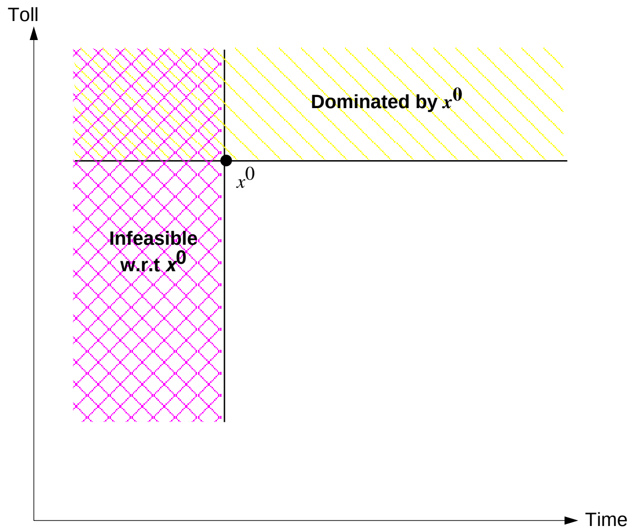

Figure 2: The Initial Solution

With respect to updating Z, we wish to begin with a *pessimistic* value to preclude overlooking any potential solutions. We can accomplish this by initially ignoring the toll constraint and solving the resulting minimum time path problem. This solution, which we denote by x^0, is illustrated in Figure 2. Since x^0 is the minimum time path, there are no feasible solutions to the left of x^0. In addition, all solutions above and to the right of x^0 are dominated by x^0 in the sense that they either have a larger time (and hence a larger composite cost) or a larger toll (again, and a larger composite cost). Thus, we need only consider solutions below and to the right of x^0.

In order to generate such solutions we consider the following problem:

$$\begin{array}{rrcl} \min_x & t^\top x & & \\ \text{s.t.} & Ax & = & b \\ & \tau^\top x & < & Z^j \\ & x & \in & \{0,1\}^n. \end{array} \tag{9}$$

That is, we want to look for a series of solutions that have tolls less than that on x^0 (since we know that they will have a larger time).

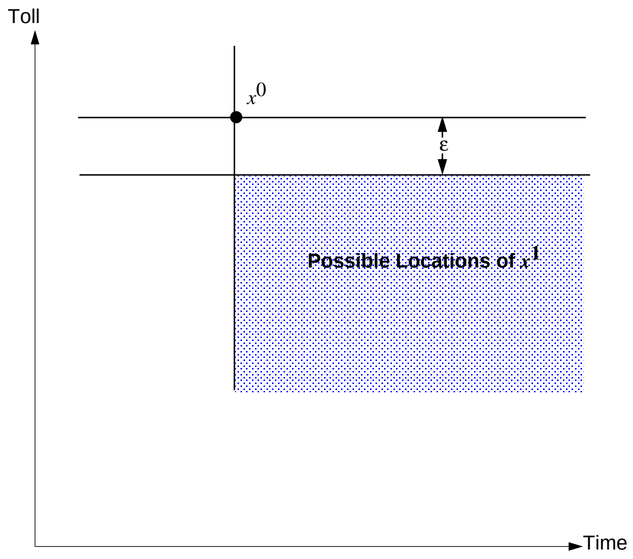

Figure 3: Updating the Constraint

Of course, we cannot incorporate a strict inequality constraint directly. Instead, as illustrated in Figure 3 (for $j = 0$), we solve:

$$\begin{array}{rrcl} \min_x & t^\top x & & \\ \text{s.t.} & Ax & = & b \\ & \tau^\top x & \leq & Z^j - \epsilon \\ & x & \in & \{0,1\}^n. \end{array} \tag{10}$$

The process iterates as shown in Figure 4, producing minimum time paths under increasingly strict toll constraints inherited from the previous iteration; each solution is said to "dominate" other paths in the restricted search space with respect to time. When a path is found having a toll equal to the minimum toll on

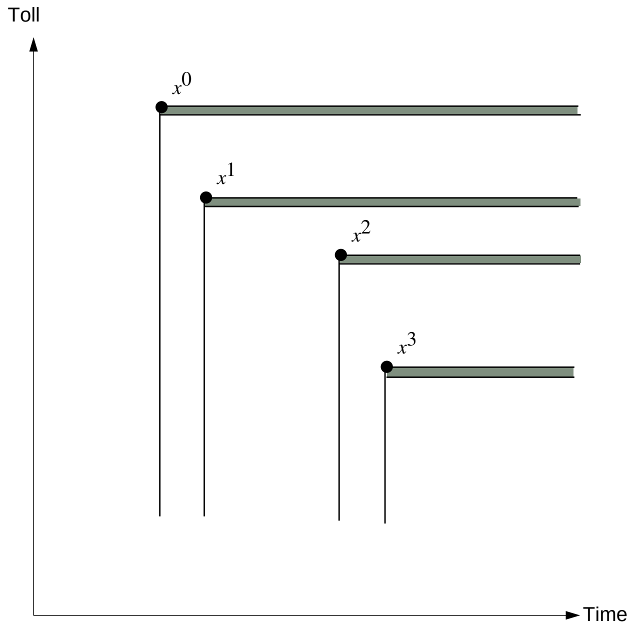

Figure 4: ICA Iterations

the network, we can terminate the algorithm and evaluate the objective function in problem (5) using each of the candidates in order to select the optimal solution. We use CSP_j to refer to the constrained problem given in (10), and let ICA denote the iterative solution process.

In practice, we can of course select an ϵ less than the minimum difference in tolls on the network (e.g., less than a penny), and hence can ensure that the search method does not overlook a potential solution when solving any of the constrained problems.

IMPROVING THE PERFORMANCE

It is possible that even the implicit enumeration of dominant paths described above can take a great many iterations. In many cases, however, we can dramatically improve the method's performance by recognizing that it may be possible to terminate the search early. This entails constructing the set of time–toll pairs that have the same composite cost as x^j, which we denote by:

$$I(x^j) = \{(a,b) : v(a) + b = v\left(t^\top x^j\right) + \tau^\top x^j, a \geq t^\top x^j, b \geq \tau^\top y^0\} \qquad (11)$$

where $\tau^\top y^0$ is the minimum toll on all paths between the origin and destination.

Since the minimum toll may not be zero, we find y^0 by solving the following minimum toll path problem:

$$\begin{array}{rrcl} \min_x & \tau^\top x & & \\ \text{s.t.} & Ax & = & b \\ & x & \geq & 0. \end{array} \qquad (12)$$

Next, let $\bar{t}(x^j)$ denote the point in $I(x^j)$ with the maximum time. That is, let:

$$\bar{t}(x^j) = \max\{a : (a,b) \in I(x^j) \text{ for some } b\}. \qquad (13)$$

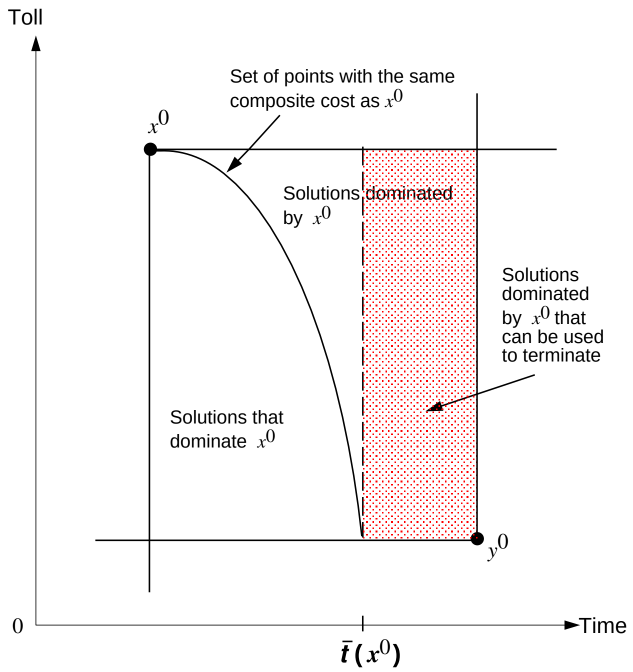

Figure 5: Terminating the Algorithm

It follows that if we obtain a solution x^{j+1} with $t^\top x^{j+1} \geq \bar{t}(x^j)$ we can terminate the algorithm (since x^{j+1} dominates all subsequent candidate solutions with respect to time). This is illustrated in Figure 5 for x^0.

Of course, it may be the case that this technique does not reduce the search space at all. Specifically, consider what happens if $\bar{t}(x^j) \geq t^\top y^0$, as shown in Figure 6. It turns out that in such cases it may be possible to work from y^0 to reduce the search space.

To do so, we formulate the "mirror" problem to CSP_j, where now we repeatedly solve a minimum toll problem subject to a time constraint. For all networks of practical interest, we can *a priori* select a lower bound for the smallest difference in path times and hence select an appropriate value of ϵ to use in our (time-)constrained shortest path problems.

With respect to bounding this search, we can again construct the set of time-toll pairs with the same composite cost as the current solution, y^k:

$$J(y^k) = \{(a, b) : v(a) + b = v\left(t^\top y^k\right) + \tau^\top y^k, a \geq t^\top x^0, b \geq y^0\} \tag{14}$$

and define a corresponding termination criterion, $\overline{\tau}(y^k)$, where

$$\overline{\tau}(y^k) = \max\{a : (a, b) \in J(y^k) \text{ for some } a\}. \tag{15}$$

We must note that, at any given iteration, the search space may not be reduced by either method (that is, all subsequent solutions may lie below and to the "left" of the inner composite cost curve). Therefore, the worst-case complexity of the inherited constraint algorithm equals that of the brute-force method (NP-hard).

In summary, the algorithm we propose for solving the nonadditive minimum cost path problem is as follows:

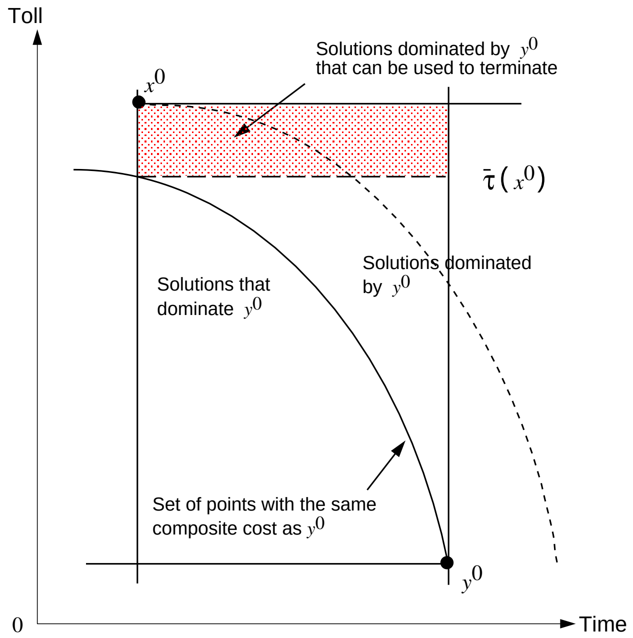

Figure 6: Developing a Termination Criterion Using y^0

Initialize:

Set $j = k = 0$.

Find y^0 and x^0. If $\tau^\top x^0 = \tau^\top y^0$, STOP – x^0 is the optimal solution.

Iterate:

while (MORE2DO) {

if ($v(t^\top x^j) + \tau^\top x^j \leq v(t^\top y^k) + \tau^\top y^k$) {

Set $j = j + 1$
Set $Z^j = \tau^\top x^{j-1}$. If ($Z^j == \tau^\top y^0$) { MORE2DO = false; break; }
Solve the subproblem CSP$_j$ for x^j

}

else {

Set $k = k + 1$
Set $W^k = t^\top y^{k-1}$. If ($W^k == t^\top x^0$) { MORE2DO = false; break; }

Solve the mirror subproblem for y^k

}

if ($t^\top x^{j+1} \geq \bar{t}(x^j)$ or $\tau^\top y^{k+1} \geq \overline{\tau}(y^k)$) "MORE2DO" = false;

}

Select optimal solution:
Calculate composite cost for each of paths enumerated and select the minimum cost path.

SOLVING THE SUBPROBLEM

Problem (10) is a simple example of a constrained shortest path (CSP) problem [see (5) for a classification of general CSP problems].

Unfortunately, the presence of the toll constraint destroys the integrality property of the constraint matrix, thus problem (10) must explicitly include the integer-value constraint $x \in \{0,1\}^n$. To solve this problem, then, we turn to Lagrangian Relaxation [see (6), (7), and (8), along with Handler and Zang (9)].

Consider the following relaxation of the problem given in (10):

$$\begin{array}{rrcl} \min_x & t^\top x + \lambda[\tau^\top x - (Z^j - \epsilon)] & & \\ \text{s.t.} & Ax & = & b \\ & x & \geq & 0 \end{array} \tag{16}$$

Here the toll constraint is introduced into the objective function with a multiplier, λ, which essentially penalizes violations of the constraint. It is well-known that, for any value of λ, the solution to this type of relaxed problem serves as a lower bound to the objective function value in the original problem. Our goal, then, is to find the λ and corresponding path x that provide the best lower bound on the optimal solution (ideally, the optimal path itself). Rewritten one last time, the problem we wish to solve is:

$$\begin{array}{rrrcl} \max_\lambda & \min_x & (t + \lambda\tau)^\top x - \lambda(Z^j - \epsilon) & & \\ & \text{s.t.} & Ax & = & b \\ & & x & \geq & 0 \end{array} \tag{17}$$

While not guaranteed to be the optimal solution of the corresponding CSP$_j$, the solution to this maximization problem is relatively easy to find since problem (17) is concave in λ and λ is a scalar. Thus we can find the optimal value of λ using a one-dimensional line search algorithm, albeit one that does not use derivatives since (17) is piecewise-linear. Furthermore, for each test value of λ and Z^j fixed, (17) is a simple linear programming problem that can be solved in a variety of ways [see (13)].

In practice, we can apply the relaxation technique using an "appropriate" value for the interval of uncertainty. A very small interval would ensure that we find a close approximation to the optimal value of λ, however, a larger tolerance is desirable from the point of reducing the number of shortest path calls made for each test λ.

Regardless of the size of the uncertainty interval chosen, some computational savings will result from using the value of λ found in the previous iteration to serve as lower bound for the line search in the current iteration. Furthermore, we can find an upper bound for the line search interval by solving the minimum toll path problem and solving the maximization problem using the minimum toll as the constraint. (Should the algorithm begin to work the mirror problem, a new upper bound must be found.)

NUMERICAL EXAMPLES

We implemented the inherited constraint algorithm in C and, using the network shown in Figure 7, examined the performance of the solution approach. Our choice of one-dimensional optimization algorithm is the Golden Section Method; we use a binary-heap implementation of Dijkstra's algorithm [(10), (11), (12)] to solve the related shortest path subproblems. The uncertainty interval used when maximizing the Lagrangian is 0.0001.

The test network represents the New Jersey highway system, and is comprised of 321 nodes and 1128 arcs (including 110 tolled arcs). We used a value of time function $v(t) = 10t^2$; the time cost associated with each arc is a function of its length and assumed travel speed. Table 1 lists the categories of roads and the travel speed assumed for each type.

Twenty representative "trips" on the network with varying origins and destinations were selected for study. Table 2 lists these origin-destination (OD) pairs, along with the cost on the corresponding best nonadditive path.

For each of these OD pairs, we first solved the nonadditive shortest path problem using the Lagrangian relaxation heuristic to solve the CSPs. In all but one of the cases (i.e., OD Pair 15), the minimum time path was determined to be the optimal path overall, which is not surprising given the value of time function used. Table 3 provides a comparison of the CPU time and number of CSPs and shortest

Figure 7: New Jersey Network

Type	Number	Speed (mph)
Ordinary highway	670	30
Divided highway	190	40
Freeway	158	50
Toll road	104	50
Toll bridge or tunnel	6	25

Table 1: Road Types and Assumed Speeds

path calls required to find the optimum path with and without invoking the early termination criteria.

We then used a k-best path method (14) to generate paths until the minimum time-minimum toll space had been spanned or the 200th best path was calculated. Using the number of shortest path calls as the metric, Table 4 contrasts the performance of the relaxation heuristic and the k-best path method.

Based on these initial results, the successive relaxation method appears to be very efficient, with or without invoking the early termination mechanism. At most 175 shortest path calls were required to solve each problem vs the worst case of 2^{110} such calculations. For most of the test problems, relatively few shortest path calculations were done and, using a Silicon Graphics Indy workstation with a MIPS R5000 CPU running at 180 MHz and 96 Mb of memory, less than 0.3 seconds of CPU time was needed with the early termination method invoked.

In contrast, when the k-best path algorithm was used, the results were mixed. In some cases (i.e., OD pairs 6, 11, 13 and 14), only 2 shortest path calls were required. (The ICA could be reconfigured to find both the minimum time and minimum toll paths before beginning any line searches, which would yield the same result.) However, several hundreds — if not thousands — of shortest path calls were completed in most of the cases tested without spanning the time-toll space and hence ensuring that the optimum path was among the paths calculated. (Note: a $>$ next to the number of shortest path calls indicates that the k-best method was terminated unsuccessfully.)

CONCLUSION AND FUTURE RESEARCH

In this paper we have demonstrated that the inherited constraint algorithm (ICA) can efficiently solve the minimum cost path problem when path costs include both time and tolls and the valuation of travel time is nonlinear.

OD Pair	Origin	Destination	Dollar Cost on Best Path
1	Paramus	Atlantic City	133.21
2	Princeton	Cape May	149.01
3	Tewksbury	Atlantic City	144.73
4	Camden	Jersey City	61.31
5	Del. Memorial Bridge	Bayonne	95.85
6	Princeton	Vernon	57.98
7	Princeton	Atlantic City	81.95
8	Princeton	Jersey City	22.64
9	Newark	Pt. Pleasant	28.48
10	Newark	Lambertville	34.72
11	Gum Tree Corner	Camden	31.44
12	Newark	Long Branch	24.77
13	Gum Tree Corner	Atlantic City	60.69
14	Princeton	Morristown	14.50
15	Paramus	Trenton	42.82
16	Paramus	Pt. Pleasant	39.64
17	New York City	Atlantic City	135.39
18	New York City	Trenton	39.16
19	Princeton	Meadowlands	23.59
20	Del Mem. Bridge	Princeton	55.21

Table 2: OD Pairs Tested

Much work remains to be done, however. Directions for future research include further validation and refinement of the algorithm, and possible extensions to the nonadditive path costs model.

Using branch and bound techniques, we wish to validate the convergence of the ICA on larger, more realistic networks (e.g., derived from TIGER files). Second, we wish to address the issue of *a priori* determination of the interval of uncertainty used in the 1-D optimization for a given OD pair. In addition, it may be that the value of time function, v, is piecewise-linear (or is well-approximated using a piecewise-linear function). We wish to consider alternative algorithms for solving the nonadditive problem in this context.

Other remaining research objectives include providing an example that realizes the worst-case, non-polynomial complexity of the nonadditive shortest path problem and laying the groundwork for more efficient implementation of the inherited constraint algorithm. An exhaustive performance evaluation is required using additional networks.

Finally, we wish to extend the work presented in this paper by incorporating the solution algorithm into a [static] model of traffic equilibrium.

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. Ahuja, R.K., T.L. Magnanti, and J.B. Orlin. *Network Flows Theory: Algorithms and Applications* , Prentice-Hall, New Jersey, 1993.

3. Hensher, D.A. and T.P. Truong. "Valuation of Travel Times Savings," *Journal of Transport Economics and Policy*, pp. 237-260, 1985.

4. Bellman, R. *Dynamic Programming*, Princeton University Press, Princeton, 1957.

5. Beasley, J.E. and N. Christofides. "An Algorithm for the Resource Constrained Shortest Path Problem," *Networks*, Vol. 19, pp. 379-394, 1989.

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

7. Fisher, M.L. "The Lagrangian Relaxation Method for Solving Integer Programming Problems," *Management Science*, Vol. 27, No. 1, pp. 1-18, 1981.

8. Shapiro, J.F. *Mathematical Programming: Structures and Algorithms*, Wiley, New York, 1979.

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

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

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

12. Tarjan, R.E. *Data Structures and Network Algorithms*, CBMS 44, SIAM, Pennsylvania, 1983.

13. Scott, K.A., G. Pabón-Jiménez, and D. Bernstein, "Finding Alternatives to the Best Path," paper no. 970682 presented at TRB, January 1997.

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

OD Pair	w/ ET CPU Time (secs)	w/ ET CSPs	w/ ET SPs	w/o ET CPU Time (secs)	w/o ET CSPs	w/o ET SPs
1	0.25	2	75	0.57	6	175
2	0.17	1	44	0.24	2	63
3	0.17	1	45	0.26	2	64
4	0.24	2	69	0.38	2	113
5	0.16	1	49	0.31	3	97
6	0.09	0	25	0.09	0	25
7	0.17	1	45	0.35	4	95
8	0.13	1	47	0.26	3	91
9	0.13	1	50	0.25	3	100
10	0.10	1	40	0.10	1	40
11	0.03	0	25	0.03	0	25
12	0.12	1	51	0.24	3	102
13	0.04	0	25	0.04	0	25
14	0.04	0	25	0.04	0	25
15	0.15	2	65	0.15	2	65
16	0.12	1	50	0.33	4	125
17	0.26	2	75	0.61	6	175
18	0.10	1	47	0.20	3	91
19	0.10	1	45	0.14	2	64
20	0.07	1	40	0.07	1	40

Table 3: Effect of Early Termination

OD Pair	Number of Shortest Path Calls	
	Relaxation Heuristic	k-Best Algorithm
1	175	>5188
2	63	>5017
3	64	>4390
4	113	>5002
5	97	>5261
6	25	2
7	95	>3746
8	91	>2998
9	100	>3246
10	40	>2244
11	25	2
12	102	1815
13	25	2
14	25	2
15	65	>3775
16	125	>4324
17	175	>5141
18	91	>3669
19	64	>2647
20	40	>3746

Table 4: Comparison of Heuristic and k-best Method