Nonadditive Shortest Paths

Steven A. Gabriel
The George Washington University

David Bernstein
Princeton University

April 1999

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

1 Introduction

The problem of finding the shortest path in a network arises frequently in transportation applications. When building descriptive behavioral models one frequently assumes that the people choose the shortest, fastest, or cheapest path. When building prescriptive systems (e.g., in-vehicle navigation systems) one almost always wants to provide a user with the "best" path to her/his destination. It is not surprising, therefore, that researchers have devoted a considerable amount of time and effort to studying this problem.

Interestingly, most of the research on shortest path problems, particularly as they relate to transportation applications, has assumed that the "cost" on a path is the sum of the "costs" on the arcs in that path. Unfortunately, however, this is often not the case. A particularly important example arises when people consider more than one attribute (e.g., travel time and tolls) and one of the attributes is valued nonlinearly (e.g., time). This situation can arise both in descriptive models (e.g., trying to predict the impact of a new toll policy) and in prescriptive systems (e.g., providing guidance when there are tolled and untolled alternatives).

In this paper we consider a shortest path problem in which the costs are a particular nonlinear, nonseparable function. We consider a specific functional form which converts arc travel times to costs via a certain "value" function (usually assumed convex) and also adds arc tolls. Recently, Bernstein and Scott [4] have developed an algorithm to solve this form of the problem and have reported good results. In this paper, we present an algorithm of the **feasible-directions variety** in which subproblems are simple minimum cost flow linear programs. The algorithm contains a heuristic component to ensure that a path which produces a lower function value is found. Based on our series of tests, the results are very favorable. Indeed, the heuristic component was never needed for convergence of the method. In addition, the total number of iterations needed (where each iteration corresponds to solving a transportation linear programming problem) was very low– typically only two or three iterations were needed–for a variety of starting points, functions, and networks considered.

The problem considered in this paper is closely related to the multicriteria path problem discussed in [6], [12], and elsewhere. Indeed, it appears [11] that the polynomial time approximation schemes that have been used to solve the multicriteria problem (see, for example, [14]). Here, however, we are not concerned with identifying the entire efficient frontier and, hence, approach the problem somewhat differently.

2 Problem Formulation

We consider a network composed of arcs $\mathcal{A}$ and nodes $\mathcal{N}$ with cardinalities $|\mathcal{A}| = n$ and $|\mathcal{N}| = m$. We distinguish two nodes, n_o and n_d as the origin and destination nodes of this network. In addition, there is a function $c : R^n_+ \to R_+$ which associates a cost with the vector of arc flows $x \in R^n_+$.

2.1 The Nonadditive Shortest Path Problem

The shortest path problem we examine can be stated as the following nonlinear, binary program:

$$\begin{array}{ll} \text{min} & c(x) \\ \text{s.t.} & Ax = b \\ & x \in \{0,1\}^n \end{array} \tag{1}$$

where A is the $m \times n$ node-arc incidence matrix, that is:

$$a_{ij} = \begin{cases} 1 & \text{if arc } j \text{ is directed out of node } i \\ -1 & \text{if arc } j \text{ is directed into node } i \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

and $b \in R^m$ is defined as:

$$b_i = \begin{cases} 1 & i = n_o \\ -1 & i = n_d \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

It is not hard to see that the feasible set $F^{IP} = \{x : Ax = b, x \in \{0,1\}^n\}$ defines paths from the origin to the destination. We also introduce the following related problem as a relaxation of (1):

$$\begin{array}{ll} \text{min} & c(x) \\ \text{s.t.} & Ax = b \\ & 0 \leq x \leq 1 \end{array} \tag{4}$$

with corresponding feasible region $F = \{x : Ax = b, 0 \leq x \leq 1\}$ and vertex set $V(F)$. The first (well-known) result concerns the relationship between the two feasible regions F^{IP} and F.

Lemma 1 *Consider the nonadditive shortest path problem described by (1) and (4). If we assume that* *(i) the origin and destination are connected by a directed path from the origin to the destination, and*

(ii) $c(x)$ is continuous,

then,

(a) problems (1) and (4) always have solutions, and

(b) the vertices of the feasible region to (4) correspond to paths in the network.

Proof. The feasible regions to both problems are closed and bounded hence compact. They are also nonempty by assumption (i). By assumption (ii), and the Weierstrass Theorem we see that both problems have a solution. Part (b) follows from Theorems 1,2 and 3 in [9].QED

2.2 A Class of Nonadditive Cost Functions

It is often assumed in shortest path problems, that the path costs are additive (i.e., they are simply the sum of the individual arcs costs for the arcs on the path). By contrast, we consider path cost functions c that are nonadditive (i.e., the costs are not simply the sum of the relevant arc costs). As discussed above, such problems often arise as subproblems in nonlinear, competitive multi-agent network models.

In its most general form, this formulation is less mathematically tractable than the additive model. However, we consider a particular variant of the nonlinear formulation as described below. Specifically, we consider the case in which:

$$c(x) \quad = v(t^T x) + \tau^T x, \tag{5}$$

where $t \in R^n_+$ denotes one component of the overall cost (which we will refer to as the vector of times), $\tau \in R^n_+$ denotes another component of the overall cost (which we will refer to as the vector of tolls), and $v : R_+ \to R_+$ is a translation function that converts time units (e.g., minutes) to toll units (e.g., dollars).

This functional form is easy to motivate if one considers the specific case of time-cost and money-cost. To do so, one need only assume that users have a nonlinear value of time. Is this reasonable? Consider the following thought experiment. Would you rather have 30 one-minute “blocks” of time or 1 thirty-minute “block” of time? If you do not place the same dollar value on these two options then you have a nonlinear value of time.

With this cost function the relaxed problem of interest to us becomes:

$$\begin{array}{ll} \text{min} & v(t^T x) + \tau^T x \\ \text{s.t.} & Ax = b \\ & 0 \leq x \leq 1 \end{array} \tag{6}$$

A basic result concerning the cost function $c(\cdot)$ is now described.

Lemma 2 *If the translation function $v(\cdot)$ is convex, then the path cost function $c(x)$ in (5) is convex.*

2.3 Motivating the Heuristic

As discussed by [4], there is a problem that is closely related to (6) that can be very useful when trying to solve (1). Specifically, let x^* denote a solution to (4) and consider the problem:

$$\begin{array}{ll} \text{min} & \sum_i \tau_i x_i \\ \text{s.t.} & Ax = b \\ & t^T x = t^T x^* \\ & 0 \leq x \leq 1 \end{array} \tag{7}$$

The following (obvious) result summarizes the important relationships between (6) and (7).

Lemma 3 *Let x^* be a solution to (6) and let $\bar{x}$ be a solution to (7). Then x^* is also a solution to (7) and $\bar{x}$ is also a solution to (6).*

Proof. If $\bar{x}$ solves (7), then $A\bar{x} = b, 0 \leq \bar{x} \leq 1$ so that $\bar{x}$ is feasible to (6). Also,

$$\begin{aligned} v(t^T \bar{x}) + \tau^T \bar{x} &= v(t^T x^*) + \tau^T \bar{x} \\ &\leq v(t^T x^*) + \tau^T x^* \end{aligned}$$

so that $\bar{x}$ solves (6).

Now we show that every solution x^* to (6) solves (7) as well. We have

$$\begin{aligned} v(t^T x^*) + \tau^T x^* &\leq \min\{v(t^T x) + \tau^T x : Ax = b, 0 \leq x \leq 1\} \\ &\leq \min\{v(t^T x) + \tau^T x : Ax = b, 0 \leq x \leq 1, t^T x = t^T x^*\} \\ &= v(t^T x^*) + \min\{\tau^T x : Ax = b, 0 \leq x \leq 1, t^T x = t^T x^*\} \\ &= v(t^T \bar{x}) + \tau^T \bar{x} \end{aligned}$$

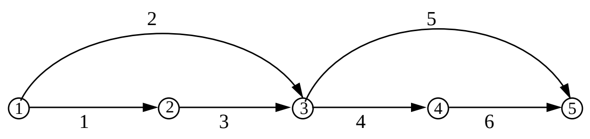
so that x^* solves (7). QED

Table 1: Times and Tolls for the Sample Network

Arc i	t_i	τ_i
1	1	0
2	1	50
3	1	0
4	1	0
5	1	50
6	1	0

The heuristic that we describe in the next section is motivated by an observation that we made after solving several instances of (6). Specifically we found that in many instances the solution set of (6) corresponds to a a path or a path with just one cycle. Indeed, the heuristic that follows seems to perform well in practice because this casual observation often holds true. However, it is relatively easy to see that this oberavtion is not true in general.

Figure 1: A Sample Network



To do so, consider the simple network shown in Figure 1 with times and tolls as shown in Table 1. The corresponding node-arc incidence for this network is:

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & -1 \end{pmatrix}$$

and the cost function for this example is $c(x) = 10(t^T x)^2 + \tau^T x$.

Solving this problem yields:

$$
\begin{aligned}
x_{12} &= 0.249967\\
x_{13} &= 0.750033\\
x_{23} &= 0.249967\\
x_{34} &= 0.250033\\
x_{35} &= 0.749967\\
x_{45} &= 0.250033
\end{aligned}
$$

with an objective function value of 137.5. Clearly, this solution does not correspond to either a path or a path with one cycle. Also, the least expensive path is 140 and since arc costs are nondecreasing with flows, no path or path with one cycle is better than this solution.

3 The Heuristic

In spite of the counterexample above, it turns out that in many cases one can solve (1) by considering the vertices of the feasible region F. Of course, this means that as long as one ends up with a vertex solution, the LP relaxation (6) can be considered in place of the original formulation. While it is not generally true that the minimum of a convex function will occur at an extreme point of the feasible region, with proper care, a best vertex solution can be achieved. This is the idea behind Algorithm 1 presented below.

In what follows, we make use of the following direction-finding subproblem in which $\bar{x}$ is a fixed vector and y is the vector of arc flows to be solved for and the function $c \in C^2$:

$$
\begin{array}{ll}
\min & \nabla c(\bar{x})^T(y-\bar{x})\\
\text{s.t.} & Ay = b\\
 & 0 \le y \le 1
\end{array}
\tag{8}
$$

Note that since the feasible region is compact and the objective function is linear, hence continuous, by the Weierstrass Theorem, this problem always has a solution. In addition, we know that since this is a linear program, there is always a vertex solution.

3.1 A Stopping Condition

We begin the discussion of the heuristic at the end. That is, we consider a stopping condition that allows one to conclude that the original binary programming problem has been solved.

Lemma 4 *Suppose that*

(i) *the function $c(\cdot)$ is convex and differentiable and*

(ii) *the vector $\bar{x}$ corresponds to a vertex of F.*

If y^* is an optimal solution to (8) given $\bar{x}$, with $\nabla c(\bar{x})^T(y^* - \bar{x}) \geq 0$, then $\bar{x}$ solves (1).

Proof. First note that since y^* is a solution (8), it corresponds to a path. Since c is convex and differentiable, we know that for every x in the domain of c

$$c(y) \geq c(x) + \nabla c(x)^T(y - x)$$

for all y. Using the optimality of y^* and the nonnegativity of the associated inner product term, we have

$$\begin{aligned} c(y) - c(\bar{x}) &\geq \nabla c(\bar{x})^T(y - \bar{x}), \forall y \in F \\ &\geq \nabla c(\bar{x})^T(y^* - \bar{x}) \\ &\geq 0, \end{aligned} \tag{9}$$

which implies that $c(y) \geq c(\bar{x})$ $\forall y \in V(F) \subset F$ or that $\bar{x}$ is a solution to (1). QED

3.2 How to Choose the Next Iterate

Lemma 4 specifies a stopping condition that indicates when a path vector generated from the LP subproblem corresponds to a solution to the original integer program. Now we consider when the optimal objective function value to (8) is not nonnegative as described in the lemma above. In this case we have, $\nabla c(\bar{x})^T(y^* - \bar{x}) < 0$ and the vector y^* provides a descent direction for c and also y^* corresponds to a path. This latter fact follows from Lemma 1 since y^* is an optimal solution to the subproblem and must correspond to a vertex since (8) is an LP.

To choose the next iterate, two cases need to be distinguished. First, if

$$c(y^*) < c(\bar{x}),$$

then we know that the new iterate y^* provides a lower function value than $\bar{x}$ and corresponds to a path, thus we should take y^* as the next iterate. (Even if $\bar{x}$ did not correspond to a vertex, taking y^* as the next iterate is useful for descent of the method).

On the other hand, if

$$\nabla c(\bar{x})^T(y^* - \bar{x}) < 0, \text{ and } c(y^*) \geq c(\bar{x}),$$

then in spite of the fact that $y^* - \bar{x}$ was a descent direction, if we require descent of the overall objective function, we cannot move to the vertex y^*.

3.2.1 Moving Off of a Vertex

Of course, in some cases we must move off of a vertex. One approach is to backtrack and find the largest value $\bar{\rho} \in \{\rho^0, \rho^{-1}, \rho^{-2}, \ldots\}$ such that

$$c(\bar{x} + \bar{\rho}(y^* - \bar{x})) < c(\bar{x})$$

where $\rho \in (0, 1)$. This ensures that the next iterate, $\bar{x} + \bar{\rho}(y^* - \bar{x})$, has a lower function value. Since y^* is a descent direction, such a steplength is guaranteed to exist. The difficulty is that the next iterate, $\bar{x} + \bar{\rho}(y^* - \bar{x})$, may not be a vertex. Indeed, this is exactly the reason why general descent methods for convex programs with network-type constraints will not necessarily work. (Note that in principle, one could allow the algorithm to move away from a vertex solution early on to generate sufficient descent of the objective function and then later, require that only vertex solutions be used for iterates.)

3.2.2 Searching for a Better Path Solution to the Subproblem

If there is a path corresponding to the vector $\hat{y}$ for which $c(\hat{y}) < c(\bar{x})$ then, assuming that c is convex, we would have

$$c(\hat{y}) - c(\bar{x}) \geq \nabla c(\bar{x})^T(\hat{y} - \bar{x}).$$

Note that such a path might not have been calculated from the LP subproblem because the quantity $\nabla c(\bar{x})^T(\hat{y} - \bar{x})$ might not have been optimal. That is, minimizing the directional derivative would have produced a direction that gave the *most* negative inner product only, not one in which the new direction pointed to a path (vertex) with a lower cost. In this case, one can perform a heuristic search, via a sequence of linear programming subproblems to determine if such a vector $\hat{y}$ exists or if, in fact, there is no descent in the objective function relative to moving to a new path. We discuss several heuristic procedures for this step shortly. First, however, we describe the overall heuristic.

We denote the "best path"[1] solution at a given iteration as y^{best}. "Best" in this context means that, out of all those paths considered, this path has the lowest objective function value. In the initialization step, we select a $y^{best} \in R^n_+$ with corresponding objective function arbitrarily large so that at iteration zero, the path solution $(y^*)^0$ will necessarily be a better path.

Algorithm 1: An Overall Heuristic

Step 0 (Initialization) Select a vector $x^0, y^{best} \in R^n_+$.

Set $k = 0$.

Step 1 (Search Direction)

- **(a)** Compute $(y^*)^k$ as a solution to the LP (8) with $\bar{x} = x^k$.
- **(b)** If $c((y^*)^k) < c(y^{best})$, then $y^{best} \leftarrow (y^*)^k$.

Step 2 (Calculate Iterate)

- **(a)** If $\nabla c(x^k)^T((y^*)^k - x^k) \geq 0$, then STOP, $(y^*)^k$ is an optimal solution to (1).
- **(b)** Otherwise, if $c((y^*)^k) < c(x^k)$ then set $x^{k+1} = (y^*)^k$, $k = k+1$ and go to Step 1.
- **(c)** Else, determine using a heuristic approach, if there exists a vertex $\hat{y}$ such that $c(\hat{y}) < c(y^{best})$. If yes, set $x^{k+1} = \hat{y}$, $k = k+1$, $y^{best} = \hat{y}$ and go to Step 1. If no, then stop, y^{best} solves (1).

Clearly, if Step 2c can be satisfied for each relevant iteration, this method produces a sequence of iterates $\{x^k\}$ that converge to a solution. The key is to identify good heuristics for solving the Step 2c problems.

[1] y^{best} need not actually refer to a path.

4 Heuristic Approaches for Finding a Path with Lower Cost

In Step 2 of the algorithm, we consider the case when the LP subproblem has produced an optimal solution $(y^*)^k$ corresponding to a path for which:

$$\nabla c(x^k)^T((y^*)^k - x^k) < 0.$$

but

$$c((y^*)^k) \geq c(x^k).$$

The following approaches are heuristic schemes to either find a path $\hat{y}$ such that $c(\hat{y}) < c(y^{best})$ or to determine that no such $\hat{y}$ exists.

4.1 A Path-Finding Heuristic

In this scheme, if there is a path $\hat{y}$ such that

$$c(\hat{y}) - c(y^{best}) < 0,$$

then, assuming that c is a convex function, this implies that

$$0 > \nabla c(y^{best})^T(\hat{y} - y^{best}).$$

Hence, we can make use of the following related linear program as part of the search for a better path $\hat{y}$:

$$\begin{array}{ll} \text{min} & \nabla c(y^{best})^T(y - y^{best}) \\ \text{s.t.} & Ay = b \\ & 0 \leq y \leq 1 \end{array} \qquad (10)$$

Since y^{best} was a vertex which was computed previously (ignoring the initial value), the optimal objective function of (10) must be negative. Since if it were nonnegative, by Lemma 4, we would have concluded that y^{best} was in fact an optimal solution to the original integer program. Denote by y^*, an optimal solution to (10). We wish to solve a modified form of (10) in which we attempt to generate a $\hat{y}$ by forcing the LP to satisfy $\nabla c(y^{best})^T(y - y^{best}) \geq \nu$ where ν is a value in the range $(\nabla c(y^{best})^T(y^* - y^{best}), 0)$. More specifically, if we denote the heuristic inner iteration by i, at the ith iteration, we would solve the following LP:

$$\begin{array}{ll} \min & \nabla c(y^{best})^T(y - y^{best}) \\ \text{s.t.} & Ay = b \\ & 0 \leq y \leq 1 \\ & \nabla c(y^{best})^T(y - y^{best}) \geq \nu_i \end{array} \tag{11}$$

Note that this LP always has a solution since the feasible region is compact and nonempty (since y^{best} satisfies the constraints) and the objective function is continuous. Moreover, since y^{best} is a feasible solution, the objective function is bounded above by 0. Note that this approach has a similarity to the feasible directions method of Zoutendijk (see, for example, [2]). The main difference being that in our method, we solve a sequence of these LP subproblems to find a solution that corresponds to a vertex (i.e., a path). This is not necessarilly the case in the method of Zoutendijk.

If the constraint involving ν_i is binding at an optimal solution, then the solution may not refer to a vertex. However, we know from examining the KKT optimality conditions, that if this constraint is not tight, then a vertex solution to (11) corresponds to a vertex solution from (8).

In any event, even if this new constraint is binding, we can take the computed solution and easily check to see if it relates to a path. It is worth noting that one can also apply Lagrangian relaxation techniques (see, for example, [10, 16]) to (11). Specifically, letting:

$$\begin{array}{lll} L(\lambda) = & \min & \nabla c(y^{best})^T(y - y^{best}) - \lambda(\nabla c(y^{best})^T(y - y^{best}) - \nu_i) \\ & \text{s.t.} & Ay = b \\ & & 0 \leq y \leq 1 \end{array} \tag{12}$$

one can maximize $L(\lambda)$ in an attempt to solve (11). Alternatively, one can simply select a large value of λ and hope to get lucky.

In what follows, y^* is a solution to (8) used in the overall heuristic.

Algorithm 2: A Path-Finding Heuristic

Step 0 (Initialization) Select a value for $\nu_0 \in (\nabla c(y^{best})^T(y^* - y^{best}), 0)$. Set $i = 0$. Set $tol_\nu > 0$, $\delta > 0$.

Step 1 (Termination Check for ν) If $|\nu_i| \leq tol_\nu$ then no $\hat{y}$ exists.

Step 2 (Candiate Path Generation) Solve the LP (11) denoting a solution as $(y^*)^i$. If $(y^*)^i$ does not relate to a path, then go to Step 3c.

Step 3 (Termination Check for $\hat{y}$)

- **(a)** If $\nabla c(y^{best})^T((y^*)^i - y^{best}) \geq 0$, then no $\hat{y}$ exists, STOP.
- **(b)** Else, if $c((y^*)^i) < c(y^{best})$ then set $\hat{y} = (y^*)^i$ and STOP.
- **(c)** Otherwise, select a $\nu_{i+1} \in (\nu_i, 0)$, with $\nu_{i+1} - \nu_i \geq \delta$, set $i = i + 1$ and go to Step 1.

It is important to note that in Step 2, if $(y^*)^i$ does not relate to a path, we simply increase the value of ν_i and repeat. Thus, without loss of generality we can assume that all solutions to (11) correspond to a path. After possibly readjusting for small enough (integral) values for t and τ (for example seconds and pennies, respectively), the term $\nabla c(y^{best})^T((y^*)^i - y^{best})$ can only take on certain prescribed values. This is because, the term

$$\nabla c(y^{best}) = v'(t^T y^{best})t + \tau$$

has a fixed nonnegative value and the term $(y^*)^i - y^{best}$ is the difference of two path vectors. Thus, we know for example that if we denote $w = \nabla c(y^{best})$, that

$$-\sum_i w_i \leq \nabla c(y^{best})^T((y^*)^i - y^{best}) \leq \sum_i w_i$$

so that a possible range for ν could be $(-\sum_i w_i, 0)$. The lowest amount to advance ν by would be 0.01 if the units of t, τ, and v were properly adjusted.

Lemma 5 *In a finite number of iterations (specifically $\frac{tol_v - \nu_0}{\delta}$), Algorithm 2 will either find a less costly path than y^{best} or determine that no such path exists.*

Proof. Suppose not. Then there is an infinite sequence of ν_i values generated from Step 3c. However, since

$$\nu_0 \in (\nabla c(y^{best})^T((y^*) - y^{best}), 0)$$

and $\nu_{i+1} - \nu_i > \delta$, the maximum number of iterations is

$$\frac{tol_v - \nu_0}{\delta} < \infty,$$

and thus we have the desired result. QED

5 Numerical Results

The Algorithm 1 presented above has been applied to two reasonably-sized networks under a variety of choices for the value function $v(\cdot)$, starting points, and (O,D) pairs. The results are very encouraging in that the method converged to a solution in all cases in only 2 or 3 iterations. That is to say, a shortest path to problem (1) was computed by solving only 2 or 3 associated linear programming subproblems. In fact, in each case, the method always produced a vertex solution as the next iterate and the heuristic scheme as described above to ensure this was not needed.

At this time the exact reasons why this was the case are not well-known and further research in this area is needed. Indeed, we can construct a simple counter-example to show that maintaining each iterate as a vertex does not necessarily always happen. Consider the following simple counter-example.

Counter-Example

We consider the network shown in Figure 2 with three nodes and three arcs given as $\{(1,2),(1,3),(2,3)\}$ with the corresponding node-arc incidence matrix given as follows:

$$A = \begin{pmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & -1 \end{pmatrix}.$$

Taking the origin as node 1 and the destination as node 3 we see that there are just two possible paths: $1 \to 2 \to 3$ (path 1), or $1 \to 3$ (path 2). Suppose that the times on each of the three arcs are 1 and the tolls are respectively, $(0,23,0)$ for arcs $\{(1,2),(1,3),(2,3)\}$. Using a value function of $v(\alpha) = 10\alpha^2$, the two paths have the following nonadditive costs:

path 1	value = 40
path 2	value = 33

But now consider the feasible arc flows flow vector (0.25,0.75,0.25). The objective value associated with this vector is 32.875, confirming that a non-vertex solution is optimal. Consequently, if the proposed algorithm is tried on this counter-example network with the data as given, the heuristic scheme to maintain a vertex solution would need to be employed. Since the heuristic scheme did not ever need to be invoked in our tests, one might conclude that in practice, things are perhaps somewhat better than anticipated.

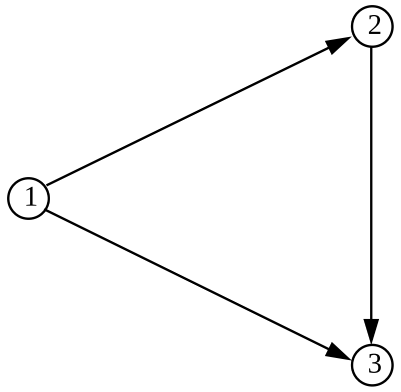


Figure 2: The Network for the Counter-Example

Description of the Test Data

The Networks

We have tested the proposed algorithm on two reasonably-sized networks. The first network, shown in Figure 3 is composed of 321 nodes and 1128 arcs representing the road network for Jersey. Since the maximum number of arcs for a network with n nodes is $n(n-1)$, we see that this New Jersey network is relatively sparse since it has only 1.1% of the maximum number of links. We have used node 152 as the origin and node 291 as the destination in our tests of this network.

The second network tested consists of 60 nodes and $3,540$ links. Since $3,540 = 60*59$ we see that this network represents the other extreme, namely a 100% dense network. The times and tolls were generated randomly for this network.

The Value Functions, the Starting Points, and the (O,D) Pairs

To provide a somewhat representative sample of value functions (that convert travel times to dollars), we have included the following three choices:

1. $v(\alpha) = 10\alpha^2$,

2. $v(\alpha) = \frac{\alpha^4}{100}$, and

3. $v(\alpha) = e^{\frac{-\alpha}{1000}}$

It is easy to check that all three choices are convex and continuously differentiable so that the optimality conditions of Lemma 4 are valid. The first two choices represent increasing functions in travel times whereas the third is decreasing.

Since an optimal arc flows vector must have each component equal to zero = or one, we have used the following choices for starting points x^0:

1. $x_i^0 = 1$, for all i,

2. $x_i^0 = 1$, for all i that are multiples of 2, $x_i^0 = 0$ otherwise,

3. $x_i^0 = 1$, for all i that are multiples of 4, $x_i^0 = 0$ otherwise,

4. $x_i^0 = 1$, for all i that are multiples of 6, $x_i^0 = 0$ otherwise,

5. $x_i^0 = 1$, for all i that are multiples of 8, $x_i^0 = 0$ otherwise.

Lastly, we have used the following choice of O-D pairs for each network:

The results of our tests appear below. Note that 'Iter' is the number of sub-problem iterations needed (i.e., the number of linear programs solved).

Here is the optimal path with associated values:

Table 1: Optimal Path for the New Jersey Network, $v(\alpha) = 10\alpha^2$=

Arc	From	To	Arc Time	Arc Toll
529	152	151	0.08975	0.00000
525	151	154	0.06719	0.00000
538	154	168	0.07071	0.00000
590	168	166	0.08883	0.00000
583	166	254	0.05340	0.00000
915	254	159	0.03802	0.00000
557	159	256	0.07106	0.30000
926	256	259	0.20839	0.40000
934	259	260	0.05701	0.30000
939	260	290	0.07159	0.30000
1044	290	289	0.03606	0.25000
1038	289	291	0.04924	0.30000

Table 2: Results for the New Jersey Network, $v(\alpha) = 10\alpha^2$

x^0	Iter	$c(x^0)$	$c(x^*)$
1	2	232,384.24	9.97
2	2	57,612.29	9.97
3	2	13,510.00	9.97
4	2	7,502.60	9.97
5	2	3,538.74	9.97

Table 3: Optimal Path for the New Jersey Network, $v(\alpha) = \frac{\alpha^4}{100}$

Arc	From	To	Arc Time	Arc Toll
527	152	148	0.10035	0.00000
513	148	146	0.22883	0.00000
505	146	145	0.10000	0.00000
504	145	172	0.06988	0.00000
605	172	173	0.10753	0.00000
610	173	247	0.24447	0.00000
892	247	262	0.17180	0.00000
946	262	263	0.05893	0.00000
950	263	264	0.06719	0.00000
954	264	284	0.07169	0.00000
1019	284	285	0.06067	0.00000
1022	285	286	0.08498	0.00000
1026	286	288	0.06719	0.00000
1035	288	291	0.04123	0.30000

Table 4: Results for the New Jersey Network, $v(\alpha) = \frac{\alpha^4}{100}$

x^0	Iter	$c(x^0)$	$c(x^*)$
1	3	5,379,912.78	0.35
2	3	329,617.97	0.35
3	3	18,068.52	0.35
4	3	5,574.42	0.35
5	3	1,254.13	0.35

Table 5: Optimal Path for the New Jersey Network, $v(\alpha) = e^{\frac{-\alpha}{1000}}$

Arc	From	To	Arc Time	Arc Toll
527	152	148	0.10035	0.00000
513	148	146	0.22883	0.00000
505	146	145	0.10000	0.00000
504	145	172	0.06988	0.00000
605	172	173	0.10753	0.00000
610	173	247	0.24447	0.00000
892	247	262	0.17180	0.00000
946	262	263	0.05893	0.00000
950	263	264	0.06719	0.00000
954	264	284	0.07169	0.00000
1019	284	285	0.06067	0.00000
1022	285	286	0.08498	0.00000
1026	286	288	0.06719	0.00000
1035	288	291	0.04123	0.30000

Table 6: Results for the New Jersey Network, $v(\alpha) = e^{\frac{-\alpha}{1000}}$

x^0	Iter	$c(x^0)$	$c(x^*)$
1	2	448.36	1.30
2	2	219.93	1.30
3	2	109.46	1.30
4	2	112.27	1.30
5	2	7.33	1.30

Acknowledgments

We would like to thank Professor Hernan Abeledo of George Washington University and Professor Jong-Shi Pang of The Johns Hopkins University for useful discussions concerning this paper.

References

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

[2] M. S. Bazaraa and C. M. Shetty, *Nonlinear Programming Theory and Algorithms*, John Wiley & Sons, New York, (1979).

[3] D. Bernstein and S. A. Gabriel, "Solving the Nonadditive Traffic Equilibrium Problem" *Lecture Notes in Economics and Mathematical Systems Network Optimization Conference*, P. M. Pardalos, D. W. Hearn, W. W. Hager, eds., (1997), 72–102.

[4] D. Bernstein and K. Scott, "Solving the Minimum Cost Path Problem When the Value of Time Function is Nonlinear," Princeton University (1997), Working Paper. .

[5] V. Chvátal, *Linear Programming*, W. H. Freeman and Company, New York, (1980).

[6] R.B. Dial, "A Model and Algorithm for Multicriteria Route-Mode Choice," *Transportation Research*, Col. 13B, (1979) 311–316.

[7] M. Frank and P. Wolfe, "An Algorithm for Quadratic Programming," *Naval Research Logistics Quarterly*, Vol. 3, (1956), 95–110.

[8] S. A. Gabriel and D. Bernstein, "The Traffic Equilibriuim Problem with Nonadditive Costs,"*Transportation Science*, Vol. 31, No. 4, (1997),337–348.

[9] R.S. Garfinkel and G.L. Nemhauser, *Integer Programming*, Wiley-Intersciences, (1972).

[10] A. Geoffrion, "Lagrangian Relaxation for Integer Programming," *Mathematical Programming Study*, Vol. 2, (1974), 82–114.

[11] M. Marathe and R. Jacob, Los Alamos National Laboratory, Personal Communication.

[12] P. B. Mirchandani and M. M. Wiecek, "Routing with Nonlinear Multiattribute Cost Functions," *Applied Mathematics and Computation*, Vol. 54, (1993), 215–239.

[13] R. T. Rockafellar, *Network Flows and Monotropic Optimization*, John Wiley & Sons, (1984).

[14] A.R. Warburton, “Approximation of Pareto Optima in Multiple-Objective Shortest-Path Problems,” *Operations Research*, Vol. 35, (1987), 70-79.

[15] J. G. Wardrop, “Some Theoretical Aspects of Road Traffic Research,” *Proc. Inst. Civil Engineers* Part II 1 (1952) 325–378.

[16] G.Y. Handler and I. Zang, Handler, G.Y, and I. Zang. “A Dual Algorithm for the Constrained Shortest Path Problem,” *Networks*, Vol. 10, (1980), 293–310.

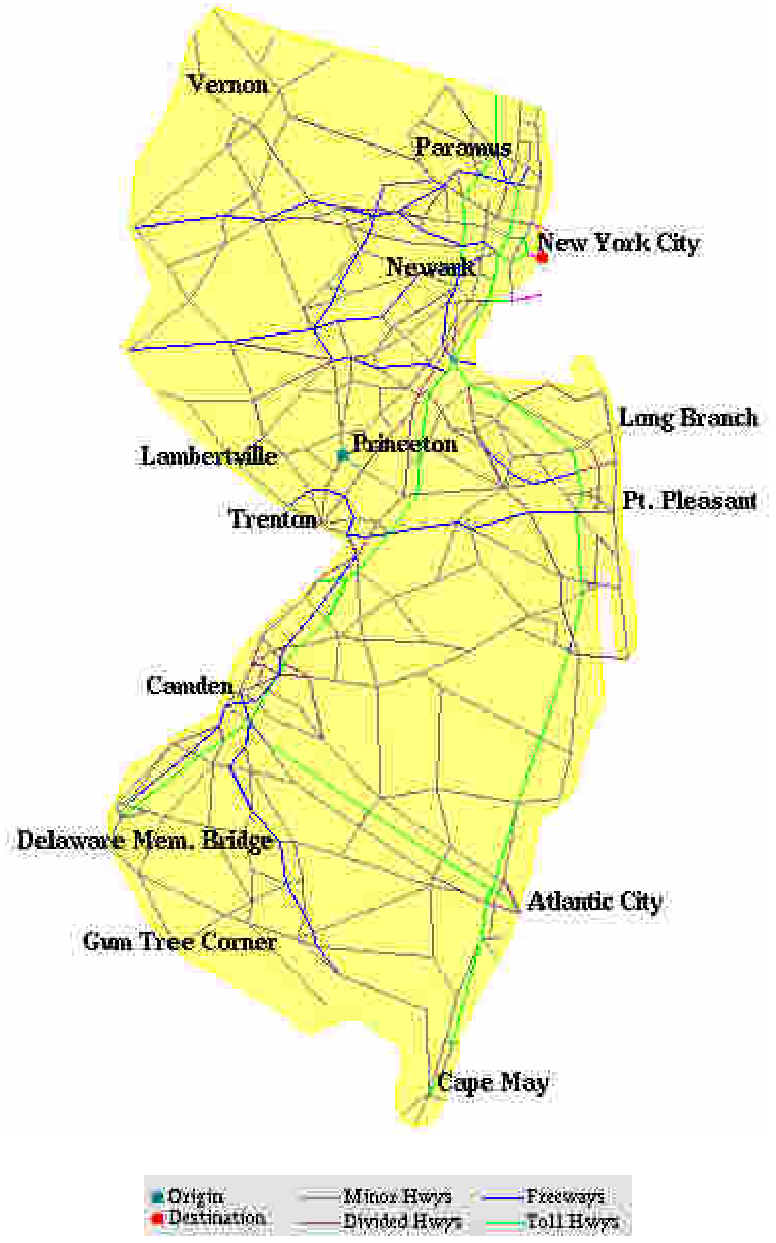


Figure 3: The New Jersey Highway Network