Mathematical Programming 53 (1992) 45-61
North-Holland

On an instance of the inverse shortest paths problem

D. Burton

Belgian National Fund for Scientific Research, Department of Mathematics, Facultés Universitaires ND de la Paix, B-5000 Namur, Belgium

Ph.L. Toint

Department of Mathematics, Facultés Universitaires ND de la Paix, B-5000 Namur, Belgium

Received 10 August 1989
Revised manuscript received 5 March 1990

The inverse shortest paths problem in a graph is considered, that is, the problem of recovering the arc costs given some information about the shortest paths in the graph. The problem is first motivated by some practical examples arising from applications. An algorithm based on the Goldfarb–Idnani method for convex quadratic programming is then proposed and analyzed for one of the instances of the problem. Preliminary numerical results are reported.

Key words: Graph theory, shortest paths, inverse problems, quadratic programming.

1. Introduction

The best way to introduce the inverse shortest paths problem is probably by its following application, drawn from mathematical traffic modelling.

In this field of applied mathematics, it is often assumed that the users of a given network of roads tend to use the shortest route from the origin of their trip to its destination, the cost of a trip being evaluated in time, distance, money or some other more complex measure. The road network planners are obviously extremely interested in the repartition of the traffic flow along those shortest routes. They also have an a priori measure of the cost of a given arc in the network, and hence they could compute the shortest routes quite easily, using one of the well developed algorithms for this problem (see [4, 5, 14], for instance). However, the precise assessment of the cost of a route (in the user's mind) is complex, and often different from that used by the planners. It is therefore very useful to know some of the routes that are actually used, and then to incorporate this knowledge into the model, modifying the a priori costs to guarantee that the given route is indeed shortest in the modified network. Care must also be exercised in avoiding large changes in the costs compared to their a priori values.

This is an instance of the inverse shortest paths problem. One is given a directed graph and a set of nonnegative costs on its arcs. The question is to modify these costs as little as possible to ensure that some given paths in the graph are shortest paths between their origin and destination.

Another interesting example is in seismic tomography (see [9, 10, 13, 15]). The network represents a discretization of the geologic zone to study into a large number of "cells", and the costs of the arcs represents the transmission time of certain seismic waves from one cell to the next. Earthquakes are then observed, and the arrival time of the resulting seismic perturbations is recorded at various observation stations on the surface. The question is to reconstruct the transmission times between the cells from the observation of shortest time waves and a priori knowledge of the geological nature of the zone under study.

Obviously, the list of applications is far from being exhaustive: the determination of the internal transmission properties of an inaccessible zone from outside measurements is a very common preoccupation in many scientific fields. But we believe that, because of their practical importance, the two examples above are enough to motivate the study of the inverse shortest paths problem.

The authors are unaware of other methods specifically designed for solving inverse shortest paths problems. As a consequence, no comparison will be presented in the section relative to numerical experiments.

2. The inverse shortest paths problem

More formally, we describe the problem as follows.

We define a weighted oriented graph as the triple $(\mathcal{V}, \mathcal{A}, c)$, where $(\mathcal{V}, \mathcal{A})$ is an oriented graph with n vertices and m arcs, and where c is a set of nonnegative costs $\{c_i\}_{i=1}^m$ associated with the arcs. We denote the vertices of $\mathcal{V}$ by $\{v_k\}_{k=1}^n$ and the arcs of $\mathcal{A}$ by $\{a_j = (v_{s(j)}, v_{t(j)})\}_{j=1}^m$, with $s(j)$ being the index of the vertex at origin of the jth arc and $t(j)$ the index of the vertex at its end.

We assume that such a weighted oriented graph $G = (\mathcal{V}, \mathcal{A}, \bar{c})$ is given, together with a set of acyclic paths

$$p_j = (a_{j_1}, a_{j_2}, \ldots, a_{j_{l(j)}}) \quad (j = 1, \ldots, p), \tag{1}$$

where $l(j)$ is the number of arcs in the jth path (its length), and where

$$t(j_i) = s(j_{i+1}) \quad \text{for } i = 1, \ldots, l(j) - 1. \tag{2}$$

If we define $\bar{c}$ as the vector in the nonnegative orthant of $\mathbb{R}^m$ whose components are the given initial arc costs $\{\bar{c}_i\}$, the problem is then to determine c, a new vector of arc costs, such that

$$\min_c \|c - \bar{c}\| \tag{3}$$

is achieved under the constraints that

$$c_i \geqslant 0 \quad (i = 1, \ldots, m), \tag{4}$$

and that the paths $\{p_j\}_{j=1}^{p}$ are shortest paths in $G^+ = (\mathcal{V}, \mathcal{A}, c)$.

Clearly, a number of interesting variants of this basic problem can be constructed by considering various norms in (3). In particular the ℓ_1, ℓ_2 and ℓ_∞ norms seem attractive. In this paper, we will restrict ourselves to the ℓ_2 norm, or least squares norm, mostly because it is widely used and leads to tractable computational methods. Other choices will be examined in future work. One could also modify the problem by introducing other objective functions of the $\{c_i\}$ to minimize. These objective functions may be linear, quadratic or generally nonlinear. Investigation of these alternatives is again deferred to further research.

As a consequence, we can rewrite the problem (3) as

$$\min_{c_i} \tfrac{1}{2} \sum_{i=1}^{m} (c_i - \bar{c}_i)^2 \tag{5}$$

subject to (4) and the p shortest path constraints. These last constraints may be expressed as a (possibly large) set of linear constraints of the type

$$\sum_{k \mid a_k \in q} c_k \geqslant \sum_{k \mid a_k \in p_j} c_k \quad (j = 1, \ldots, p), \tag{6}$$

where q is any path with the same origin and destination as p_j. As a consequence, the set of feasible costs, $\mathcal{F}$ say, is convex as it is the intersection of a collection of half space. The problem of minimizing (5) subject to (4) and (6) is then a classical quadratic programming (QP) problem. This QP is, however, quite special because its constraint set is (potentially) very large[1], very structured, and possibly involves a nonnegligible amount of redundancy. Also the problem of minimizing (5) on the set $\mathcal{F}$ of feasible costs may be considered as the computation of a projection of the unconstrained minimum onto the convex set $\mathcal{F}$. Again, the special structure of $\mathcal{F}$ distinguishes this problem from a more general projection.

3. Algorithm design

3.1. *The Goldfarb-Idnani method for convex quadratic programming*

The algorithm we present below is a specialization of the dual QP method by Goldfarb and Idnani [7]. The idea of this method is to compute a sequence of optimal solutions to the quadratic programming problems involving only some of the constraints that are present in the original problem, that is a sequence of *dual feasible points.* An *active set* of constraints is maintained by the procedure, that is, a set of constraints which are binding at the current stage of the calculation. A new violated constraint is incorporated into this set at every iteration of the procedure (some other constraint may be dropped from it), and the objective function value

[1] In general, the number of constraints can be exponential.

monotonically increases to reach the desired optimum. This approach was chosen for two main reasons:

- Since the Goldfarb–Idnani (GI) algorithm is a dual method, it is extremely easy to incorporate new constraints once a first solution has been computed. In our context, this means that, if a new set of prescribed shortest paths is given, modest computational effort will be required to update the solution of the problem.
- The GI method has an excellent reputation for efficiency, especially in the case where the number of constraints is large and near-degeneracy very likely. In particular, the method avoids slow progress along very close extremal points of the constraint set $\mathscr{F}$.

Also, the GI method and its efficient implementation are discussed in the literature, by Goldfarb and Idnani in their original paper, but also by Powell [11, 12], for example.

Because our method heavily relies on the GI algorithm, we now state this method in its full generality. In this form, it is designed for solving the QP problem given by

$$\begin{aligned} &\min_x \qquad f(x) = a^{\mathrm{T}}x + \tfrac{1}{2}x^{\mathrm{T}}Gx, \\ &\text{subject to} \quad E_i(x) \overset{\text{def}}{=} n_i^{\mathrm{T}}x - b_i \geqslant 0 \quad (i = 1, \ldots, h), \end{aligned} \tag{7}$$

where x, a and $\{n_i\}_{i=1}^h$ belong to $\mathbb{R}^m$, G is an $m \times m$ symmetric positive definite matrix, b is in $\mathbb{R}^h$ and the superscript T denotes the transpose. As indicated above, the GI algorithm maintains a set of currently active constraints, A say, and relies on the matrix N whose columns are the normals n_i of the constraints in the active set A. The matrix N is thus of dimension $m \times |A|$, where $|A|$ is the number of constraints in A. The algorithm also uses two additional matrices, namely

$$N^* \overset{\text{def}}{=} (N^{\mathrm{T}}G^{-1}N)^{-1}N^{\mathrm{T}}G^{-1}, \tag{8}$$

which is the Moore–Penrose generalized inverse of N in the space of variables under the transformation $y = G^{1/2}x$, and

$$H \overset{\text{def}}{=} G^{-1}(I - NN^*), \tag{9}$$

which is the reduced inverse Hessian of the quadratic objective function in the subspace of points satisfying the active constraints. Using these notations, the GI algorithm may now be stated as follows (see [7]):

Step 0. Find the unconstrained minimum.

Set $x \leftarrow -G^{-1}a$, $f \leftarrow \frac{1}{2}a^{\mathrm{T}}x$, $H \leftarrow G^{-1}$, $A \leftarrow \emptyset$ and $u \leftarrow (0)$.

Step 1. Choose a violated constraint, if any.

Compute the constraint value $\{E_i(x)\}_{i=1}^h$. If all constraints are satisfied, the current x is the desired solution. Otherwise, a violated constraint is chosen, that is, an index q is selected in $\{1, \ldots, h\}$ such that $E_q(x) < 0$. Also set

$$u^+ \leftarrow \begin{cases} \begin{pmatrix} u \\ 0 \end{pmatrix} & \text{if } |A| > 0, \\ 0 & \text{if } |A| = 0. \end{cases} \tag{10}$$

Step 2. Compute the primal and dual step directions.

These directions are computed by the relations

$$d = Hn_q, \tag{11}$$

and if $|A| > 0$,

$$r = N^* n_q. \tag{12}$$

Step 3. Determine the maximum steplength to preserve dual feasibility.

Define

$$S = \{j \in \{1, \dots, |A|\} \mid r_j > 0\}. \tag{13}$$

The maximal steplength that will preserve dual feasibility is then given by

$$t_f = \begin{cases} \dfrac{u_l^+}{r_l} = \min\limits_{j \in S} \left[\dfrac{u_j^+}{r_j}\right] & \text{if } S \neq \emptyset, \\ +\infty & \text{otherwise.} \end{cases} \tag{14}$$

Step 4. Determine the steplength to satisfy the qth constraint.

This steplength is only defined when $d \neq 0$, and is then given by

$$t_c = -\frac{E_q(x)}{d^T n_q}. \tag{15}$$

Step 5. Take step and update the active set.

If $t_f = \infty$ and $d = 0$, then the original QP (7) is infeasible and the algorithm stops with suitable message.

Otherwise, if $d = 0$, update the Lagrange multipliers by

$$u^+ \leftarrow u^+ + t_f \begin{pmatrix} -r \\ 1 \end{pmatrix} \tag{16}$$

and drop the lth constraint, that is $A \leftarrow A \backslash \{l\}$, where l has been determined in (14). Then go back to Step 2 after updating H and N^*.

If $d \neq 0$, t_c is well-defined, and one sets

$$t = \min[t_f, t_c], \tag{17}$$

$$x \leftarrow x + td, \tag{18}$$

$$f \leftarrow f + t(\tfrac{1}{2}t + u^+_{|A|+1})\, d^T n_q \tag{19}$$

and

$$u^+ \leftarrow \begin{cases} u^+ + t\begin{pmatrix} -r \\ 1 \end{pmatrix} & \text{if } |A| > 0, \\ u^+ + t & \text{if } |A| = 0. \end{cases} \tag{20}$$

If $t = t_c$, then set $u \leftarrow u^+$, add constraint q, that is $A \leftarrow A \cup \{q\}$, and go back to Step 1 after updating H and N^*. If, on the other hand, $t = t_f$, drop the lth constraint, that is $A \leftarrow A \backslash \{l\}$ and go back to Step 2 after updating H and N^*.

Note that u, the vector of Lagrange multipliers, has a dimension equal to the number of active constraints.

We observe that the GI algorithm involves three types of possible iterations:

1. The first is when the new violated constraint is linearly independent from those already in the active set, and all the active constraints remain active at the new solution of the QP subject to the augmented set of constraints. This occurs when $t = t_c$.

2. The second is when the new violated constraint is linearly dependent on those already in the active set. This occurs when $d = 0$ or, equivalently, when $Nr = n_q$. In order to preserve independence of the active set (that is, linear independence of the columns of N), an old constraint (the lth) is dropped from the set before incorporating the new one. As a result, N is always of full column rank.

3. The third is when the solution of the QP subject to the augmented set of constraints is such that one of these constraints is not binding. This occurs when $t = t_f$, in which case the lth constraint ceases to be binding. As one wishes to keep only binding constraints in the active set, this constraint is dropped.

We refer the reader to [7] for further details on the general GI algorithm, and in particular for the proof that it indeed solves the QP (7), provided a solution exists.

Our purpose, in the next paragraphs, is to specialize the GI algorithm to the inverse shortest paths problem given by (5), (4) and (6). We will therefore examine the successive stages of the algorithm presented above, where the structure of the problem allows some refinement.

3.2. Constraints in the active set

We first wish to analyze how to detect the violation of constraints (6), as required in Step 1.

3.2.1. Shortest paths constraints

For each of the given paths p_j, we first define P_j as the set of vertices in V that are attained by this path, that is

$$P_j \overset{\text{def}}{=} \{s(a_{j1}), t(a_{j1}), t(a_{j2}), \ldots, t(a_{j,l(j)})\}. \tag{21}$$

The vertex $s(a_{j1})$ is called the *origin* or *source* of the jth path, and denoted s_j. For every such path p_j with source s_j and for a given vector c of arc costs, it is then possible to compute all the shortest paths in $(\mathcal{V}, \mathcal{A}, c)$ from the source s_j to all the other vertices of P_j. We will then detect a violated constraint if, for some vertex $v \in P_j \setminus \{s_j\}$, one has that the predecessor of v on the shortest path from s_j to v is different from the predecessor of v in the path p_j.

In this situation, it is easy to verify that there must be a vertex $w \in P_j$ closest to v (possibly s_j), such that w is also on the shortest path from s_j to v. Furthermore, there exist two distinguished paths from w to v, the first one, noted I^+, being the shortest path, and the second one, noted I^-, being given as a subpath of p_j. The set of both these paths is called a *violating island*, and is denoted by I. The path I^+ is called its *positive shore*, while I^- is called its *negative shore*. Furthermore, the

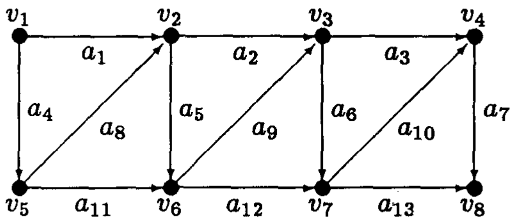

Fig. 1. A first example.

excess of the island, denoted by E, is defined as the cost of the positive shore minus the cost of the negative shore. The constraint associated with the island I is therefore violated when its excess is negative.

On the small example given in Figure 1, we assume that the cost vector c is given by the relation $c_j = j$ (that is the arc a_j has a cost of j), while the constraint paths are given by

$$p_1 = (a_1, a_5, a_{12}, a_{13}) \quad \text{and} \quad p_2 = (a_{11}, a_{12}, a_{10}). \tag{22}$$

At this point, it is not difficult to verify that the shortest path from v_1 to v_8 is the path

$$(a_1, a_2, a_3, a_7). \tag{23}$$

Hence a constraint related to the path p_1 is violated at the vertex v_8, because the predecessor of v_8 on its shortest path from v_1, that is v_4, is different from its predecessor on the constraint path, which is v_7. The vertex v above is then v_8, while inspection shows that the relevant vertex w is v_2. The corresponding violating island is then

$$I = ((a_2, a_3, a_7), (a_5, a_{12}, a_{13})), \tag{24}$$

where $I^+ = (a_2, a_3, a_7)$ is its positive shore, $I^- = (a_5, a_{12}, a_{13})$ its negative shore, and whose associated excess E is $(2+3+7)-(5+12+13) = -18$. This violating island, is not the only one for this example. A second one, related to the path p_2, is given, for instance, by

$$I' = ((a_8, a_2, a_3), (a_{11}, a_{12}, a_{10})), \tag{25}$$

whose excess E' is equal to -20.

A violated constraint of the type (6) therefore corresponds to a violating island in $(\mathcal{V}, \mathcal{A}, c)$. When it is incorporated in the active set, the constraint is enforced as an equality and the costs of its negative and positive shore are exactly balanced (see Section 3.4). The corresponding island is then called *active.*

3.2.2. Nonnegativity constraints and bounds on the arc costs

The nonnegativity constraints (4) must also be taken into account. When one of them is violated, which is easy to detect, it may also be incorporated in the active set, along with the active islands. These bounds are then also called *active.* They will be regarded in the sequel as active islands with only one arc in the positive shore and no negative shore.

The active set at a given stage of the calculation will therefore contain a number of active islands (with or without negative shore). This will be denoted by $A=(V, Y)$, where V is the set of currently active islands with a negative shore and Y the set of active islands without negative shore, that is the set of active bounds.

3.3. The dual step direction

The next stage of the specialization of the Goldfarb-Idnani algorithm to our inverse shortest paths problem is the computation of the dual step direction r in (12). As in [7, 12], this calculation, which is equivalent to

$$r=(N^{\mathrm{T}}G^{-1}N)^{-1}N^{\mathrm{T}}G^{-1}n_q, \tag{26}$$

can be performed by maintaining a triangular factorization of the matrix $N^{\mathrm{T}}G^{-1}N$. However, our problem has the very important feature that the Hessian matrix G of the quadratic objective is the identity I. This obviously induces a number of useful algorithmic simplifications, the first one being that (26) can be rewritten as

$$r=(N^{\mathrm{T}}N)^{-1}N^{\mathrm{T}}n_q. \tag{27}$$

The matrix N^* is then nothing but the unweighted Moore-Penrose generalized inverse of N. Therefore, we will only maintain a triangular factorization of the form

$$N^{\mathrm{T}}N=R^{\mathrm{T}}R, \tag{28}$$

where R is a upper triangular matrix of dimension $|A|$. Since N is of full rank, this is equivalent to maintaining a QR factorization of N of the form

$$N=(Q_1 \quad Q_2)\begin{pmatrix} R \\ 0 \end{pmatrix} \overset{\mathrm{def}}{=} QU, \tag{29}$$

as is the case in the numerical solution of unconstrained linear least squares problems. Indeed, it is straightforward to verify that (27) may be reformulated as

$$\min_r \|Nr-n_q\|_2. \tag{30}$$

The second useful simplification due to the special structure of the problem arises in the computation of the product $N^{\mathrm{T}}n_q$ in (27). The resulting vector indeed contains in position i the inner product of the ith active constraint normal with the normal to the qth constraint. As both these constraints may be interpreted as islands, the question is then to compute the inner product of the new island, corresponding to the qth constraint, with all already active islands. We then obtain the following simple result:

Lemma 1. *The vector $N^{\mathrm{T}}n_q$ appearing in (27) is given componentwise by*

$$[N^{\mathrm{T}}n_q]_i=|I_j^+\cap I_q^+|+|I_j^-\cap I_q^-|-|I_j^+\cap I_q^-|-|I_j^-\cap I_q^+| \tag{31}$$

for $i=1,\ldots,|A|$ and j equal to the index of the ith active island.

Proof. Since

$$[N^T n_q]_i = n_i^T n_q \tag{32}$$

it is useful to note that, because of (4) and (6),

$$[n_l]_k = \begin{cases} +1 & \text{if } a_k \in I_l^+, \\ -1 & \text{if } a_k \in I_l^-, \\ 0 & \text{otherwise,} \end{cases} \tag{33}$$

for $k = 1, \ldots, m$ and $l \in A \cup \{q\}$. This equation holds for both types of islands (with or without negative shore). Taking the inner product of two such vectors (for $l = j$ and $l = q$) then yields (31). □

As a consequence, the practical computation of r may be organized as follows:

1. Compute the vector $y \in \mathbb{R}^{|A|}$ whose ith component is given by (31).
2. Perform a forward triangular substitution to solve the equation

$$R^T z = y \tag{34}$$

for the vector $z \in \mathbb{R}^{|A|}$.

3. Perform a backward triangular substitution to solve the equation

$$Rr = z \tag{35}$$

for the desired vector r.

This calculation will be a very important part of the total computational effort per iteration in the algorithm.

3.4. Determination of the costs

We now examine the way in which changes in the costs may be computed. In the original GI method, both primal and dual step directions are computed once a new constraint has been selected for inclusion in the active set (as described in Step 2). In our framework, the computation of the new values of the primal variables may be completely deferred after that of the dual step in a rather simple way, as will now be shown. This adaptation may be viewed as another consequence of the fact that $G = I$ for our problem.

Before stating this result more precisely, we introduce some more notation. In order to complete the description of the set $\{1, \ldots, m\}$ given an active set $A = (V, Y)$, we recall the definition of Y as

$$Y \overset{\text{def}}{=} \{i \in \{1, \ldots, m\} \mid c_i = 0\}, \tag{36}$$

and we define the sets

$$X \overset{\text{def}}{=} \{i \in \{1, \ldots, m\} \setminus Y \mid \exists j \in V, a_i \in I_j\} \tag{37}$$

and

$$Z \overset{\text{def}}{=} \{1, \ldots, m\} \setminus (X \cup Y). \tag{38}$$

The set X thus contains the indices of the arcs that are involved in one of the active islands of V but are not fixed at their lower bounds. The set Z contains the indices of the arcs that are not involved at all in the active constraints of A.

For $i \in X$, we also define

$$I^+(i) \stackrel{\text{def}}{=} \{j \in V \mid a_i \in I_j^+\} \quad \text{and} \quad I^-(i) \stackrel{\text{def}}{=} \{j \in V \mid a_i \in I_j^-\}. \tag{39}$$

Hence, $I^+(i)$ (resp. $I^-(i)$) is the set of active islands of V such that the arc a_i belongs to its positive (resp. negative) shore.

We finally define the logical indicator function $\delta[\cdot]$ by

$$\delta[\textit{condition}] = \begin{cases} 1 & \text{if } \textit{condition} \text{ is true,} \\ 0 & \text{if } \textit{condition} \text{ is false.} \end{cases} \tag{40}$$

We can now state our lemma.

Lemma 2. *Consider a dual feasible solution for the problem of minimizing* (5) *subject to the constraints given by an active set $A = (V, Y)$. Assume furthermore that, among the Lagrange multipliers $\{u_k\}_{k=1}^{|A|}$, those associated with the active islands of V are known. Then the cost vector c corresponding to this dual solution is given by*

$$c_i = \delta[i \in X \cup Z]\bar{c}_i + \delta[i \in X]\left[\sum_{k \in I^+(i)} u_k - \sum_{k \in I^-(i)} u_k\right] \tag{41}$$

for $i = 1, \ldots, m$.

Proof. We first note that we can restrict our attention to the costs that are not at their bounds ($i \in X \cup Z$), because we know, by definition, that $c_i = 0$ for $i \in Y$. Every active island in V thus corresponds to a constraint of the form

$$\sum_{k \mid a_k \in I^+ \wedge k \notin Y} c_k - \sum_{k \mid a_k \in I^- \wedge k \notin Y} c_k = 0. \tag{42}$$

The desired expression for c_i ($i \in X \cup Z$) immediately follows from the Lagrangian equation

$$\frac{\partial L(c, u)}{\partial c_i} = 0, \tag{43}$$

where the Lagrangian function for the problem is given by

$$\begin{aligned} L(c, u) &= \tfrac{1}{2} \sum_{i \in X \cup Z} (c_i - \bar{c}_i)^2 - \sum_{k=1}^{|A|} u_k \left[\sum_{i \mid a_i \in I_k^+ \wedge i \notin Y} c_i - \sum_{i \mid a_i \in I_k^- \wedge i \notin Y} c_i\right] \\ &= \tfrac{1}{2} \sum_{i \in X \cup Z} (c_i - \bar{c}_i)^2 - \sum_{i \in X} c_i \left[\sum_{k \in I^+(i)} u_k - \sum_{k \in I^-(i)} u_k\right], \end{aligned} \tag{44}$$

where we restrict the last major sum to the set X because all other terms are zero. □

The lemma simply means that the ith cost can be obtained from $\bar{c}_i$ by adding to it all Lagrange multipliers corresponding to active islands such that a_i belongs to the positive shore of the island, and by substracting all the multipliers of active islands such that a_i belongs to the negative shore.

Consider now the computation of the primal step direction d and of the inner product $d^{\mathrm{T}}n_q$. Note first that, when (15) is reached in the algorithm, the primal step direction d is nonzero, and n_q is linearly independent from the columns of N. The value of $d^{\mathrm{T}}n_q$ is then given by the following result:

Lemma 3. *Assume the GI algorithm is applied to the inverse shortest paths problem under consideration, and that it has reached the point where equation* (15) *should be evaluated. Assume furthermore that $A = (V, Y)$ is the active set at this stage of the calculation. Then the primal step direction d is given componentwise by*

$$d_i = \delta[a_i \in I_q^+] - \delta[a_i \in I_q^-] + \delta[i \in X]\left[\sum_{k \in I^-(i)} r_k - \sum_{k \in I^+(i)} r_k\right] \tag{45}$$

for $i = 1, \ldots, m$. As a consequence,

$$d^{\mathrm{T}}n_q = 1 + \sum_{k \in I^-(q)} r_k - \sum_{k \in I^+(q)} r_k \tag{46}$$

in the case where the qth constraint is the lower bound on the qth cost, and

$$d^{\mathrm{T}}n_q = \sum_{i|a_i \in I_q^+}\left[1 + \sum_{k \in I^-(i)} r_k - \sum_{k \in I^+(i)} r_k\right] + \sum_{i|a_i \in I_q^-}\left[1 + \sum_{k \in I^+(i)} r_k - \sum_{k \in I^-(i)} r_k\right] \tag{47}$$

in the case where the qth constraint is a violating island.

Proof. We first note that d, the change in the cost c corresponding to a unit step in the dual step direction, can be viewed as the sum of two different terms $d = n_q - Nr$. The first term corresponds to the incorporation of the qth constraint in the active set and its contribution to d_i is $+1$ if a_i belongs to the positive shore of the qth island, and is -1 if a_i belongs to its negative shore. This is because the $(|A|+1)$th component of the dual step direction, corresponding to the qth constraint, is equal to $+1$. Hence we have that this first contribution is equal to

$$\delta[a_i \in I_q^+] - \delta[a_i \in I_q^-] \tag{48}$$

for the ith arc. Note that only one of the indicator functions can be nonzero in (48). The second contribution corresponds to the modifications to c_i caused by the fact that a_i may also belong to islands that are already active. In other words, the nonzero components of $-r$ have to be taken into account. The equation (41) then implies that this second contribution from the Lagrange multipliers associated with all constraints already in the active set must be equal to

$$\delta[i \in X]\left[\sum_{k \in I^-(i)} r_k - \sum_{k \in I^+(i)} r_k\right]. \tag{49}$$

Summing the contributions (48) and (49) gives (45).

Assume now that the qth constraint is a lower bound. In this case, one has that $n_q = e_q$, the qth vector of the canonical basis in $\mathbb{R}^m$. Hence the product $d^{\mathrm{T}}n_q$ is equal to d_q. Equation (41), the nonnegativity of the $\{\bar{c}_i\}_{i=1}^m$, and the fact that $c_q < 0$ imply that $q \in X$, and (46) then follows from (45). On the other hand, if the qth constraint is a violating island, the normal n_q is then given componentwise by (33) with $l = q$. Hence we obtain (47) from (45). □

3.5. Modifying the active set

The active set modifications (in Step 5 of the GI algorithm) finally require the updating or downdating of the triangular matrix R, as introduced above in (28).

Assume first that the lth constraint is dropped from the active set A. This amounts to dropping a column of N in (29), which, in turn, is equivalent to dropping a column of the upper triangular matrix R. The resulting matrix is therefore upper-Hessenberg, and a sequence of Givens plane rotation is applied to restore the upper triangular form. This technique is quite classical, and has already been used in the more general implementations of the GI method, both in [7] and [12]. The reader is referred to those papers for further details in the context of the GI algorithm, and to [8] for general information on Givens plane rotations and their practical computation.

If one now wishes to add the lth constraint to the active set, then N has one more column, namely n_q, and the resulting U in (29) then has the form

$$\begin{pmatrix} R & Q_1^{\mathrm{T}} n_q \\ 0 & Q_2^{\mathrm{T}} n_q \end{pmatrix}, \tag{50}$$

where Q_1 and Q_2 are defined in (29). Again, this matrix should be restored to triangular form, and again this can be done by premultiplying it by suitable orthogonal transformations. In fact, the only necessary modification to (50) is the premultiplication of the vector $Q_2^{\mathrm{T}} n_q$ by an orthogonal transformation T, say, such that

$$T Q_2^{\mathrm{T}} n_q = \| Q_2^{\mathrm{T}} n_q \| e_1, \tag{51}$$

where e_1 is the first vector of the canonical basis of $\mathbb{R}^{m-|A|}$. Note also that

$$Q_1^{\mathrm{T}} n_q = R^{-\mathrm{T}} N^{\mathrm{T}} n_q = z, \tag{52}$$

where z has already been computed in (35). Moreover, one has that

$$\| n_q \|^2 = \| (Q_1 \;\; Q_2)^{\mathrm{T}} n_q \|^2 = \| z \|^2 + \| Q_2^{\mathrm{T}} n_q \|^2 = \| z \|^2 + \| T Q_2^{\mathrm{T}} n_q \|^2. \tag{53}$$

Hence the updated matrix R is given by

$$R_{\mathrm{updated}} = \begin{pmatrix} R & z \\ 0 & \alpha \end{pmatrix}, \tag{54}$$

where $\alpha = \sqrt{\| n_q \|^2 - \| z \|^2}$. The updating of the triangular factor R is therefore extremely cheap to compute, mainly because of the fact that z is available from previous calculations. It is also interesting to note that, because of the equivalence between (28) and (29), the technique presented here is in fact identical to the computation of the Cholesky factor of $(N^+)^{\mathrm{T}} N^+$ using the bordering method (see

[6], for example), where $N^+ = (N \ n_q)$. A similar procedure is also used in [7] and [12].

We finally note that d, the primal step direction, is zero if and only if the residual of the problem (30) is zero, which, in turn, is equivalent to $\|n_q\| = \|z\|$. This last relation provides a possible way for testing the equality $d = 0$ without explicitly computing d.

3.6. *The algorithm*

We are now in position to describe our algorithm for solving our inverse shortest paths problem, as described by (5), (4) and (6). For this description, we use a small (machine dependent) tolerance $\varepsilon > 0$ to detect to what extent a real value is nonzero, and we define the integer $\nu = |A|$.

Step 0. Initialization.

Set $c \leftarrow \bar{c}$, $f \leftarrow 0$, $A \leftarrow \emptyset$, $\nu \leftarrow 0$ and $u \leftarrow 0$.

Step 1. Compute the current shortest paths.

For $j = 1, \ldots, p$, compute the shortest paths from s_j to every vertex in $P_j \backslash \{s_j\}$.

Step 2. Choose a violated island or exit.

Select I_q, an island whose excess E_q is negative, if any. If no such island exists, then c is optimal and the algorithm stops.

Otherwise, if $\nu = 0$, then $\alpha \leftarrow \sqrt{|I_q^+| + |I_q^-|}$ and go to Step 5.

Otherwise (that is, if $\nu > 0$) set

$$u \leftarrow \begin{pmatrix} u \\ 0 \end{pmatrix}. \tag{55}$$

Step 3. Revise the triangular factor $\mathbb{R}$.

3a. Add the previous constraint normal n_q to N. If $\nu = 1$ then set $R = (\alpha)$ and go to Step 4.

Otherwise (that is if $\nu > 1$), update the upper triangular matrix R using (54) and go to Step 4.

3b. Drop n_l from N. Remove from R the column corresponding to the lth island, and use Givens rotations to restore it to upper triangular form, as described in Section 3.5.

Step 4. Compute the dual step direction.

Compute the vectors z and r, using Lemma 1, (34) and (35). Compute also α according to

$$\alpha = \sqrt{\|n_q\|^2 - \|z\|^2}. \tag{56}$$

Step 5. Determine the maximum steplength to preserve dual feasibility.

Determine the set S according to (13), t_f (and possibly l) using (14).

Step 6. Determine the steplength to satisfy the qth constraint.

If $\alpha \leq \varepsilon$ then go to Step 7b.

Otherwise, compute t_c according to (15), and d and $d^T n_q$ as described in Lemma 3.

Step 7. Take the step and revise the active set.

7a. Compute the steplength t as in (17), set $c \leftarrow c + td$, revise f according to (19) and u using

$$u \leftarrow \begin{cases} u + t\begin{pmatrix} -r \\ 1 \end{pmatrix} & \text{if } \nu > 0, \\ u + t & \text{if } \nu = 0. \end{cases} \tag{57}$$

If $t = t_c$, set $A \leftarrow A \cup \{q\}$, $\nu \leftarrow \nu + 1$ and go to Step 1.

Otherwise (that is if $t = t_f$), set $A \leftarrow A \backslash \{l\}$, $\nu \leftarrow \nu - 1$ and go to Step 3b.

7b. If $t_f = +\infty$, then the problem is infeasible, and the algorithm stops with a suitable message.

Otherwise, update the Lagrange multipliers according to (16). Set $A \leftarrow A \backslash \{l\}$, $\nu \leftarrow \nu - 1$ and go to Step 3b.

Note that, in our current implementation of the algorithm's second step, we choose the current violated island as that whose excess is most negative. This technique appears to be quite efficient in practice.

3.7. Nonoriented arcs

An important variant of the basic problem occurs when some arcs in the graph are undirected. In this case, it is quite inefficient to replace each of these arcs by two distinct arcs of opposite orientation, because it increases both the dimension of the problem and the number of constraints. Indeed, one has to impose that the two new oriented arcs have the same cost.

Fortunately, the algorithm described above can be applied to the case where arcs are nonoriented without any modification, provided the shortest paths method used in Step 1 can handle such arcs.

3.8. Note

Similar implementation techniques have been used by Calamai and Conn for solving location problems with a related structure (see [1, 2, 3]). Their technique is, however, different from ours, and a comparison of both approaches will be examined in future work.

4. Preliminary numerical experience

In order to verify the feasibility of the above described algorith, a FORTRAN program was written and tested on an Apollo DN3000 workstation, using the FTN compiler.

We present here a set of seven typical examples extracted from a large collection of tests. The first five arise from the traffic modelling problem presented in Section 1, with graphs for two different cities. The next one is obtained on a randomly generated graph while the last one is built from the graph of a two dimensional rectangular grid. The problems' characteristics are reported in Table 1. We recall that n, m and p are the number of vertices in the graph, the number of arcs and the number of shortest path constraints respectively.

We summarize the results of the tests in Table 2, where the following symbols are used:

iter: the number of major iterations of the algorithm, that is, the number of full steps in the primal space (adding a constraint in the active set and requiring the calculation of the shortest paths and the choice of a new violated constraint).

drops: the number of islands dropped at Step 7 of the algorithm, that is, the number of minor iterations (partial and dual steps, involving only the computation of the step directions in the primal and dual space).

$|A^*|$: the number of active islands at the solution.

We note that the first of these numbers is always one larger than the sum of the two others, because one iteration is required for considering the empty active set.

Despite the limited character of these experiments, one can nevertheless observe the following points:

Table 1
The test examples

	n	m	p	Graph type	Constraint paths generation
P1	246	351	245	city 1	a tree in the graph
P2	246	351	600	city 1	all paths between a subset of the nodes
P3	246	351	6724	city 1	all paths from a node subset to another node subset
P4	822	1447	821	city 2	a tree in the graph
P5	822	1447	6806	city 2	all paths from a node subset to another node subset
P6	500	1469	100	random	randomly generated paths
P7	3600	7063	650	2D grid	all paths from one side of the grid to the other sides

Table 2
Results

	iter.	drops	$\|A^*\|$
P1	35	2	32
P2	77	17	59
P3	167	34	132
P4	246	55	190
P5	468	238	229
P6	436	54	381
P7	171	8	162

- The algorithm is relatively efficient in the sense that it does not, at least in our examples, add many constraints that are not active at the solution, with the necessity to drop them at a later stage.
- One also observes in practice that a fairly substantial part of the total computational effort is spent in calculating the necessary shortest paths in order to detect constraint violation. Choosing a set of constraint paths from a single tree induces significant savings in the determination of the most violated constraint, because only one shortest path tree is needed.

5. Complexity of the inverse shortest paths problem

During the refereeing period of this paper, an alternative formulation of the inverse shortest paths problem was communicated to the authors by S. Vavasis. Representing the cost of the shortest paths from node v_i to node v_j by the new variables $w_{i,j}$ for $i, j = 1, \ldots, n$, we may then add the constraints

$$[s(a_l) = v_k \text{ and } t(a_l) = v_j] \Rightarrow w_{i,j} \leqslant w_{i,k} + c_l, \tag{58}$$

together with the equalities

$$w_{i,i} = 0 \tag{59}$$

for all $i = 1, \ldots, n$. The constraints on the shortest paths (6) may then be rewritten as

$$w_{i,q} \geqslant c_{j_1} + \cdots + c_{j_{l(j)}} \tag{60}$$

for any path of the form (1) with $s(a_{j_1}) = v_i$ and $t(a_{j_{l(j)}}) = v_q$.

There are at most mn inequalities of type (58), n equalities of type (59) and $p \leqslant n^2$ inequalities of type (60). Hence the total number of constraints in this formulation is polynomial. As a consequence, the problem is solvable in polynomial time by an interior point algorithm.

This interesting observation is clearly of theoretical importance, but the inclusion of n^2 additional variables could generate inefficiencies in practical implementations.

6. Conclusion and perspectives

In this paper, the inverse shortest paths problem has been posed and a computational algorithm has been proposed for one of the many problem specifications, namely that where the variational criterion used is the ℓ_2 deviation from a priori known costs and where the constraints are given as a set of shortest paths and nonnegativity constraints on the costs.

The proposed algorithm has been programmed and run on a few examples, in order to prove the feasibility of the approach.

The possible extensions are many. Consideration of other norms and other types of constraint specifications are of obvious interest. The authors are presently working on extending the existing algorithm and program to handle general lower and upper bounds on the total cost of predefined or shortest paths.

Applying the techniques described in this paper to practical situations, in urban traffic models and tomography for instance, is also a challenging research area.

References

[1] P.H. Calamai and A.R. Conn, "A stable algorithm for solving the multifacility location problem involving Euclidian distances," *SIAM Journal on Scientific and Statistical Computing* 4 (1980) 512-525.

[2] P.H. Calamai and A.R. Conn, "A second-order method for solving the continuous multifacility location problem," in: G.A. Watson, ed., *Numerical Analysis: Proceedings of the Ninth Biennial Conference, Dundee, Scotland. Lectures Notes in Mathematics No 912* (Springer, Berlin-Heidelberg-New York, 1982) pp 1-25.

[3] P.H. Calamai and A.R. Conn, "A projected Newton method for l_p norm location problem," *Mathematical Programming* 38 (1987) 75-109.

[4] E.W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik* 1 (1959) 269-271.

[5] D.B. Johnson, "Efficient algorithms for shortest paths in sparse networks," *Journal of the Assocation for Computing Machinery* 24 (1977) 1-13.

[6] J.A. George and J.W. Liu, *Computer Solution of Large Positive Definite Systems* (Prentice-Hall, Englewood Cliffs, N.J. 1981).

[7] D. Goldfarb and A. Idnani, "A numerically stable dual method for solving strictly convex quadratic programs," *Mathematical Programming* 27 (1983) 1-33.

[8] G.H. Golub and C.F. van Loan, *Matrix Computations* (North Oxford Academic, Oxford, 1983).

[9] G. Neumann-Denzau and J. Behrens, "Inversion of seismic data using tomographical reconstruction techniques for investigations of laterally inhomogeneous media," *Geophysical Journal of the Royal Astronomical Society* 79 (1984) 305-315.

[10] G. Nolet, ed., *Seismic Tomography* (Reidel, Dordrecht, 1987).

[11] M.J.D. Powell, "On the quadratic programming algorithm of Goldfarb and Idnani," *Mathematical Programming Study* 25 (1985) 45-61.

[12] M.J.D. Powell, "ZQPCVX, A Fortran subroutine for convex quadratic programming," Report DAMTP/NA17, Department of Applied Mathematics and Theoretical Physics, University of Cambridge (Cambridge, UK, 1983).

[13] A. Tarantola, *Inverse Problem Theory. Methods for Data Fitting and Model Parameter Estimation* (Elsevier, Amsterdam, 1987).

[14] R.E. Tarjan, "Data structures and network algorithms," *CBMS-BSF Conference Series in Applied Mathematics* (SIAM, Philadelphia, PA, 1983).

[15] J.H. Woodhouse and A.M. Dziewonski, "Mapping the upper mantle: three-dimensional modeling of Earth structure by inversion of seismic waveforms," *Journal of Geophysical Research* 89 (B7) (1984) 5953-5986.