The inverse shortest paths problem with upper bounds on shortest paths costs.

by D. Burton[1] W.R. Pulleyblank[2] and Ph.L. Toint[3]

Report 93/03 March 3, 1993

Abstract. We examine the computational complexity of the inverse shortest paths problem with upper bounds on shortest path costs, and prove that obtaining a globally optimum solution to this problem is NP-complete. An algorithm for finding a locally optimum solution is proposed, discussed and tested.

[1] **Belgian National Fund for Scientific Research**
Department of Mathematics
Facultés Universitaires ND de la Paix
B-5000 Namur, Belgium
[2] **IBM T.J. Watson Research Center**
Yorktown Heights, USA
[3] **Department of Mathematics**
Facultés Universitaires ND de la Paix
B-5000 Namur, Belgium

Keywords : computational complexity, shortest paths, inverse problems, traffic modelling.

1 Introduction

Shortest paths problems form a fundamental part of transportation research, because they naturally arise in a large number of related optimization problems [9]. Shortest path techniques are applied in fields as diverse as traffic modelling [8] and computerized tomography [16]. Very efficient algorithms have been proposed during the last three decades to solve the shortest paths problem (see for example [1, 7, 15, 12, 2] for the single source problem).

However, models based on shortest paths do not always accurately reflect actual observations. These inaccuracies are often caused by errors in estimates of the arc weights or lengths used in the shortest path calculations. One way to overcome this difficulty and to obtain more accurate estimates of arc weights is to consider the *inverse* problem. Solving an inverse shortest paths problem consists of finding weights associated with the arcs of a network, that are as close as possible to *a priori* estimated values, and that are compatible with the observations of some actual shortest path lengths in the network. In [4], two authors proposed a first algorithm for handling a particular instance of this problem, where the observations consist of the complete description of the observed shortest paths. In [5], they extended this approach to the case where the arc weights are linearly correlated.

The number of possible and interesting variants of the inverse shortest path problem is large. Yet many applications (including tomography and some traffic modelling questions) feature a specific class of constraints in their formulation: bounds on the total weight of shortest paths between given origins and destinations. Unfortunately, only lower bounds on paths weights have been considered by the authors so far. It is the purpose of this paper to examine the more difficult case where upper bounds are present as well.

We motivate this development with two examples.

The first arises from seismic tomography. In this field, one is concerned with recovering ground layer densities from observations of seismic waves [13]. According to Fermat's principle, these waves propagate along rays that follow the shortest path in time across the earth crust. One can then measure, usually with some error, the propagation time of these rays between a known source and a known receiver. The problem is then to reconstruct the ground densities from these observations. One approach [17] uses a discretisation of the propagation medium into a network whose arcs have weights inversely proportional to the local density. In this framework, one is then faced with the problem of recovering these arc weights from the knowledge of intervals on seismic rays travel times and from *a priori* geological knowledge, the ray paths themselves remaining unknown. This is an inverse shortest paths problem with bounds on the paths' weights.

The second example is drawn from traffic modelling. In this research area, graph theory is used to create a simplified view of a road network. An elementary (and often justified) behavioural assumption is that network users choose *perceived* shortest routes for their journeys [18, 3, 14]. Although these routes might be observable, their precise description might vary across time and individuals, and their travel cost is usually subject to some estimation. This naturally provides bounds on the total time spent on shortest paths whose definition is unavailable. Recovering the perceived arc costs is an important step in the analysis of network users' behaviour. This is again a problem of the type considered in this paper.

In the next section, we formalise the problem and explain why bound constraints on shortest paths weights cannot fit into the framework of classical convex quadratic programs, as used in [4] and [5]. In Section 3, we study the computational complexity of the global solution of our problem and show that it is NP-complete. An algorithm for calculating a local solution is then proposed in Section 4. This algorithm is analysed in Section 5. Finally, some conclusions and perspectives are given in Section 6.

2 The problem

We now define the problem more formally and discuss its special nature.

Consider a directed weighted graph $(\mathcal{N}, \mathcal{A}, w)$, where $(\mathcal{N}, \mathcal{A})$ is an oriented graph with n nodes and m arcs, and where w a set of nonnegative weights $\{[w]_i\}_{i=1}^m$ associated with the arcs, where we use $[\cdot]_i$ to denote the i-th component of a vector. Let $\mathcal{N}$ be the set of nodes of the graph and $\mathcal{A} = \{a_k\}_{k=1}^m$ be the set of arcs. A *path* is then defined as a set of consecutive arcs of $\mathcal{A}$.

As presented in [4], the idea of the inverse shortest paths method is to determine the arc weights that are as close as possible to their expected values, subject to satisfying the shortest path constraints. Denoting these *a priori* expected values by $\{\bar{w}_i\}_{i=1}^m$ and choosing the ℓ_2 norm to measure the proximity of the weight vectors w and $\bar{w}$, we therefore obtain the following least squares problem

$$\min_{w \in \mathbf{R}^m} f(w) = \frac{1}{2} \sum_{i=1}^{m} ([w]_i - [\bar{w}]_i)^2 \tag{1}$$

subject to the constraints

$$[w]_i \geq 0, \quad i = 1, \ldots, m \tag{2}$$

and the bound constraints on the cost of shortest paths

$$\sum_{a \in p_q^1(w)} [w]_a \leq u_q, \quad q = 1, \ldots, n, \tag{3}$$

where $p_q^1(w)$ is the shortest path (with respect to the weights w) starting at node o_q and arriving at the node d_q[1]. The values of u_q are upper bounds on the cost of the shortest path from o_q to d_q. We allow u_q to be infinite. Note that the shortest path $p_q^1(w)$ is not necessarily unique for a given w. This will have important implications later in this paper.

The method proposed by the authors in [4] is based on the quadratic programming algorithm due to Goldfarb and Idnani [10]. The idea is to compute a sequence of optimal solutions to the problem involving only a subset of the constraints present in the original problem. The method therefore maintains an *active set* of constraints. Starting from the unconstrained solution, each iteration *incorporates* a new constraint in the active set, completing what we call a *major* iteration.

[1]The meaning of the superscript 1 in $p_q^1(w)$ is to indicate that the *shortest* path is considered, as opposed to the *second shortest*.

To achieve this goal, it may be necessary to *drop* a constraint from the the active set. These drops occur in *minor* iterations.

Incorporating upper bound constraints of the type (3) in the active set is complex. The difficulty is that expression (3) only defines the path $p_q^1(w)$ implicitly, while adding a constraint to the active set (as a linear inequality on arc weights) requires an explicit definition of that constraint of the form

$$\sum_{a \in p} [w]_a \leq u_q \tag{4}$$

where one needs the *explicit definition* of the path p as a succession of arcs to specify which arcs appear in the summation. When such a constraint is activated, one naturally *chooses* a path which is currently shortest given the value of the arc weights $[w]_a$. However, as these weights are modified in the course of the optimisation, a path that is shortest between a given origin and destination may vary, and therefore the explicit definition of the constraint in the form (4) should also vary accordingly. An immediate consequence of this observation is that, besides adding and dropping constraints of the type (4) from the active set, one should also keep track of the modifications in the explicit definitions of the constraints (3) which might in turn modify the active set.

3 The complexity of the problem

3.1 The convexity of the problem

The difficulty of handling constraints of type (3) can be partially explained by the fact that they generate a *nonconvex* feasible region. Our goal of finding a global minimiser of the objective function (1) with a method of low complexity then appears much more difficult, despite the fact that the objective is strictly convex.

We illustrate the nonconvex nature of our constraints with a small example. Consider the following graph, composed of 3 nodes and 3 arcs ($m = 3$), shown in Figure 1.

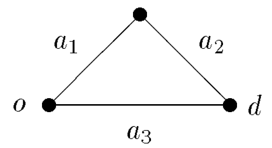

Figure 1: A small graph

Consider now the problem of minimising (1) subject to the constraint

$$\sum_{a \in p^1(w)} [w]_a \leq 5, \tag{5}$$

where $p^1(w)$ is the shortest path (with respect to the weights w) from node o and to node d. It is easy to see that $w^1 = (2 \quad 2 \quad 10)^T$ and $w^2 = (10 \quad 10 \quad 4)^T$ are feasible solutions, while $\frac{1}{2}(w^1 + w^2) = (6 \;\; 6 \;\; 7)^T$ is infeasible. The problem is therefore nonconvex.

3.2 The 3-SAT problem as an inverse shortest path calculation

As mentioned in [4], the original inverse shortest paths problem is solvable in polynomial time since an equivalent formulation of the problem contains a polynomial number of convex constraints. The original problem then belongs to the class P of decision problems solvable in polynomial time by a deterministic algorithm [11].

A problem is *NP-hard* if every problem in the class NP of problems, solvable in polynomial time on a nondeterministic Turing machine, can be transformed to it. See [11]. Cook [6] proved that there exist NP-hard problems by showing that the "satisfiability" problem has the property that every other problem in NP can be polynomially reduced to it. Therefore, if the satisfiability problem can be solved with a polynomial time algorithm, then so too can every problem in NP. In effect, the satisfiability problem is a "hardest" problem in NP. Many other combinatorial problems, such as the travelling salesman problem, have since been proved to have this same "universal" property.

The class of *NP-complete* problems consists of all such problems which belong themselves to NP.

We now show that the addition of the constraints (3) makes the inverse shortest path problem NP-hard. This we do by reducing a known NP-hard problem to it.

A particular instance of the satisfiability problem, the 3-SAT problem, is one of the best known NP-complete problems. We follow [11] for a brief description. Let X be a set of Boolean *variables* $\{x_1, x_2, \ldots, x_l\}$. A *truth assignment* for X is a function $t : X \to \{\text{true}, \text{false}\}$. Let x be a variable in X, then we say that x is realized under t if $t(x) = \text{true}$. The variable $\bar{x}$ will be realized under t if and only if $t(x) = \text{false}$. We say that x and $\bar{x}$ are *literals* defined upon the variable x. A *clause* is a set of literals over X, such as $\{x_1, x_2, \bar{x}_3\}$, representing the disjunction of those literals and is *satisfied* by a truth assignment if and only if at least one of its member is realised under that assignment. A set $\mathcal{C}$ of clauses over X is *satisfiable* if there exists some truth assignment for X that simultaneously satisfies all the clauses in $\mathcal{C}$. The 3-SAT problem consists in answering the question: is there a truth assignment for $\mathcal{C}$, when the clauses in $\mathcal{C}$ contain exactly 3 literals over X? Cook proved that this problem is NP-complete [6]. We can show that another problem is NP-hard by showing that 3-SAT can be polynomially transformed to it.

Let ISP denote the decision problem: given an inverse shortest path problem and a bound k, does there exist a solution with objective value at most k? We show that ISP is NP-complete, which implies that the inverse shortest path problem is NP-hard.

Theorem 1 *ISP is NP-complete.*

Proof. We proceed as follows:

1. show that problem ISP is in NP,
2. construct a transformation from 3-SAT to ISP, and
3. prove that this is a polynomial transformation.

The first requirement is easy to verify in our case, because the shortest path problem itself can be solved in polynomial time and all a nondeterministic algorithm solving ISP need do is guess a set of arc weights and verify in polynomial time that they satisfy the constraints.

Let us now examine the second requirement and consider a 3-SAT problem with l variables and p clauses. We represent each variable x_i by a small (sub)graph with two distinct paths between a node s_i to a node d_i (see Figure 2). The variable x_i will be true or false depending

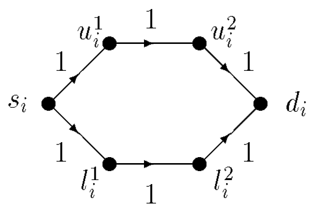

Figure 2: The representation of x_i

on whether the shortest path from s_i to d_i follows the upper path (via nodes u_i^1, u_i^2) or lower one (via nodes l_i^1, l_i^2) of its associated graph. Imposing that

$$s_i = d_{i-1} \quad (\text{for } i = 2, \ldots, l), \tag{6}$$

we obtain a "chain-like" resulting graph representing our Boolean variables. A path from node s_1 to node d_l in this graph is therefore equivalent to a truth assignment of all Boolean variables. We assign an initial cost of 1 to each of the six arcs of the "Boolean graph" x_i.

We now describe a representation of our p clauses. A clause c of the 3-SAT problem is a disjunction of type $(x_i \vee x_j \vee \bar{x}_k)$, for instance. The clause c will be associated with the choice among three possible paths going from a node named a_c to a node named b_c, where a_c and b_c are different from the nodes of the Boolean graphs x_i $(i = 1, \ldots, l)$. Each of the three paths is formed by three consecutive oriented arcs. The first arc originates at node a_c and has a zero cost, and the last one terminates at node b_c and also has zero cost. The middle arc is determined as one of the arcs (l_i^1, l_i^2) or (u_i^1, u_i^2) depending on whether the considered variable x_i in the clause is negated or not. The subgraph associated with a clause c of the type $(x_i \vee x_{i+1} \vee \bar{x}_k)$ is illustrated in Figure 3.

Our representation of the variables x_i and the clauses c_j generates an weighted oriented graph, which we call $\mathcal{G}$. The original cost of any path between a_c and b_c $(c = 1, \ldots, p)$ is 1, and the cost of the shortest path from s_1 to d_l is $3l$. The 3-SAT problem then is equivalent to the question: is there a choice of nonnegative arc weights in $\mathcal{G}$ such that the cost of the shortest path between each pair of nodes (a_c, b_c) is zero as well as that of the shortest path from s_1 to d_l, and such that the ℓ_2 distance of these weights to the original weights is at most $3l$?

The equality constraints on the shortest paths in this formulation may be replaced by upper bound constraints provided that we require the arcs weights (w) to be as close as possible to the

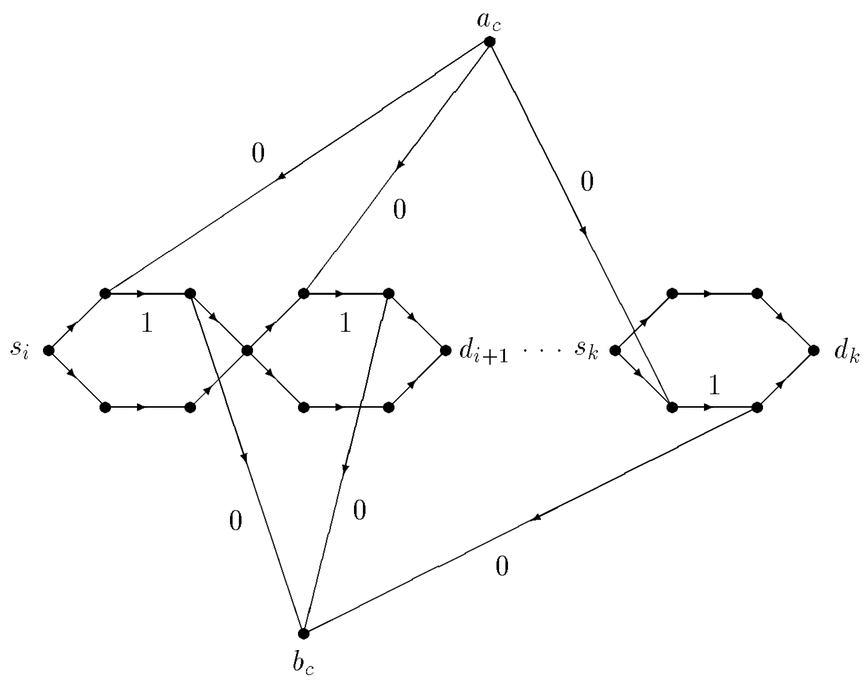

Figure 3: The subgraph associated with clause c

original weights $\bar{w}$. The resulting problem is therefore

$$\min_{w} \|w - \bar{w}\|^2 \tag{7}$$

subject to

$$w \geq 0 \tag{8}$$

$$\text{cost}(a_c, b_c) \leq 0 \quad c = 1, \ldots, p \tag{9}$$

$$\text{cost}(s_1, d_l) \leq 0, \tag{10}$$

where $\text{cost}(n_1, n_2)$ is the cost of a shortest path from n_1 to n_2 in $\mathcal{G}$.

We recognise, in the formulation (7)–(10), an instance of our inverse shortest paths problem with upper bound constraints on the cost of shortest paths. We thus have found a transformation of the 3-SAT problem to ISP.

Finally, it is easy to see that this transformation is polynomial, since the instance of ISP we constructed has $6l + 6p$ arcs and $5l + 1 + 2p$ nodes. This complete our proof that ISP is NP-complete. □

4 An algorithm for computing a local optimum

For convex problems, each critical point is a global optimum. In our nonconvex context, we shall be happy with a local minimum, that is a set of weights $\hat{w}$ such that, for all w in the intersection of the feasible domain and a neighbourhood of $\hat{w}$, one has that $f(w) \geq f(\hat{w})$.

In [4, 5], we considered a dual approach to find a global solution to the inverse shortest paths problem, because of its robustness and the insurance to reach a global optimum. Yet, this approach presents a drawback in our new context: it heavily relies on convexity.

On the other hand, primal methods typically generate a sequence of primal feasible iterates ensuring a monotonic decrease of the objective function. They do not rely so much on convexity and have the further advantage of giving a approximated solution satisfying the constraints when the iteration process is interrupted. This approach "from the inside" is the one we have chosen to follow. The general outline of our proposal is as follows.

1. We first compute a feasible starting point.
2. At each iterate, we revise the explicit definition of the shortest path constraints, and solve the resulting convex problem, using the algorithm proposed in [4].
3. The calculation is stopped when no further progress can be obtained in this fashion.

4.1 Computing a starting point

Selecting a good starting point is important in this framework. We propose to use the algorithm of [4] to find our starting weights. This algorithm will indeed compute an optimal solution of a variant of the problem where the explicit description of the path constraints is kept fixed at that chosen in $\bar{w}$. This variant is

$$\min_{w \in \mathbf{R}^m} \frac{1}{2} \sum_{i=1}^{m} ([w]_i - [\bar{w}]_i)^2 \tag{11}$$

subject to

$$[w]_i \geq 0, \quad i = 1, \ldots, m \tag{12}$$

and the bound constraints on the cost of the *a priori* shortest paths

$$\sum_{a \in p_q^1(\bar{w})} [w]_a \leq u_q, \quad q = 1, \ldots, n. \tag{13}$$

It is important to note that w_1, the solution of (11)–(13), is feasible for the original ISP, although it may not be optimal because, at w_1, the explicit definition of the shortest path constraints may differ from that at $\bar{w}$. This calculated vector is therefore a suitable starting point for a primal algorithm.

4.2 Updating the explicit constraint description

Given a set of weights w, we must *choose* an explicit description of the constraints associated with with w. Together with the quadratic objective function, this new explicit description then defines a convex quadratic program.

We emphasised the word "choose" above because there might be more than a single shortest path $p_q^1(w)$ between the origin o_q and destination d_q of the q-th original constraint in the ISP.

Denoting the number of shortest paths from o_q to d_q (for a given w) by $n_q(w)$, we define $p_q^1(w, i_j)$ ($i_j = 1, \ldots, n_q(w)$) as the i_j-th shortest path from o_q to d_q. This definition assumes that we ordered the $n_q(w)$ shortest paths, for instance using lexicographic order. For convenience, we (re)define $p_q^1(w)$ as the "first" shortest path from o_q to d_q, that is

$$p_q^1(w) \stackrel{\text{def}}{=} p_q^1(w, 1). \tag{14}$$

For future reference, the possible convex feasible regions determined, for a given w, by the constraints (2)–(3) will be denoted by $F(w, i_1, \ldots, i_n)$, where i_q ($q = 1, \ldots, n$) varies from 1 to $n_q(w)$. Again, for convenience, we define

$$F(w) \stackrel{\text{def}}{=} F(w, \underbrace{1, \ldots, 1}_{n}). \tag{15}$$

Using these notations, $P(w)$ and $P(w, i_1, \ldots, i_n)$ respectively denote the problem of minimising $f(w)$ subject to $w \in F(w)$, or in $F(w, i_1, \ldots, i_n)$. Finally, F denotes the generally nonconvex feasible domain determined by (2) and (3).

Updating the constraint description at w therefore amounts to specifying $F(w, i_1, \ldots, i_n)$ for some choice of the indices $i_1, \ldots, i_n$.

4.3 Reoptimization

Once $F(w, i_1, \ldots, i_n)$ has been determined at the feasible point w, it is possible to solve the associated convex quadratic program $P(w, i_1, \ldots, i_n)$. This process is called "reoptimisation".

Because we assume that reoptimisation will always take place at a point w which is the solution of another subproblem $P(w', i'_{,1}, \ldots, i'_n)$, it is not difficult to see that the new subproblem differs from the old one in two different ways.

1. Some constraints of $P(w', i'_{,1}, \ldots, i'_n)$ are now obsolete because the associated path, although shortest for w', is no longer shortest for w. These constraints must be replaced by constraints whose explicit description corresponds to path that are shortest for w.

2. Although $p_q^1(w', i'_1 \ldots, i_n)$ can still be shortest for w, another shortest path between o_q and d_q may be chosen to define the new subproblem. The constraint whose explicit description corresponds to $p_q^1(w', i_1, \ldots, i_n)$ must then be replaced by another constraint with explicit description corresponding to $p_q^1(w, i_1, \ldots, i_n)$.

As a consequence, some linear inequalities of the form (4) are dropped from the subproblem and some new ones are added.

Adding new linear inequalities can be handled computationally by using the Goldfarb-Idnani dual quadratic programming method, as is already the case in [4]. Removing linear inequalities can be handled much in the same way by computing the Goldfarb-Idnani step that would *add* them and then taking the opposite. Because these calculations are straightforward applications of [4], we do not describe them in detail.

4.4 The algorithm

We are now in position to specify our proposal for an algorithm that computes a local solution of ISP.

Step 0: Initialization

Compute w_1 using the inverse shortest paths algorithm of [4] for solving $P(\bar{w})$.
Set $i \leftarrow 1$ and $C_1 \leftarrow (1, \ldots, 1)$

Step 1: Update the feasible region

Compute $F(w_i, C_i)$.

Step 2: Reoptimization

Compute w_{i+1} the solution of $P(w_i, C_i)$, using the inverse shortest paths algorithm of [4].
If $w_{i+1} \neq w_i$, set $i \leftarrow i + 1$ and go to Step 1 with $C_i = (1, \ldots, 1)$.

Step 3: Choose another shortest path combination

Is there, amongst the $n_i^c = \prod_{q=1}^{n} n_q(w_i)$ possible shortest path combinations at w_i, one that has not been considered yet?
If no, stop: w_i is a local minimum of $P(w)$.
Otherwise, redefine C_i to be $(i_1, \ldots, i_n)$, the n-uple of indices corresponding to an untried combination and go to Step 1.

The reader might wonder if the (possibly costly) loop between Steps 3 and 1 is necessary. We now show that this is the case by providing a simple example, in which it is not sufficient to examine the $(1, \ldots, 1)$ combination of shortest paths only, or even to consider every possible shortest path separately.

Consider the small graph, composed of 9 nodes and 11 arcs, shown in Figure 4.

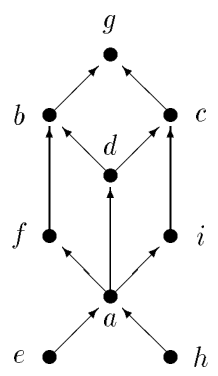

Figure 4: A small example

Let us assume that $[\bar{w}]_i = 10$ for $i = 1, \ldots, 11$, and consider the problem of minimising (1) with $m = 11$ subject to 12 constraints of type (3), defined by

$$\left\{\begin{array}{lll} o_1 = a, & d_1 = b, & u_1 = 10, \\ o_2 = a, & d_2 = c, & u_2 = 10, \\ o_3 = d, & d_3 = b, & u_3 = 5, \\ o_4 = d, & d_4 = c, & u_4 = 5, \\ o_5 = e, & d_5 = f, & u_5 = 10, \\ o_6 = f, & d_6 = g, & u_6 = 10, \\ o_7 = h, & d_7 = i, & u_7 = 10, \\ o_8 = i, & d_8 = g, & u_8 = 10, \\ o_9 = e, & d_9 = a, & u_9 = 5, \\ o_{10} = h, & d_{10} = a, & u_{10} = 5, \\ o_{11} = b, & d_{11} = g, & u_{11} = 5, \\ o_{12} = c, & d_{12} = g, & u_{12} = 5. \end{array}\right. \tag{16}$$

We directly see that, at any solution, all arcs but ad will have a weight equal to 5, since $n_q = 1$ for all $q \neq 1, 2$. Suppose now that, for these latter constraints, the shortest paths have been ordered as in (16) and have been considered by the algorithm in that order. As a consequence, solving the problem in the feasible region $F(\bar{w}, 1, \ldots, 1)$ will give the solution $[\hat{w}]_i = 5$ ($i = 1, \ldots, 11$), since the shortest path from a to b and that from a to c both use node d. The objective function value at $\hat{w}$ is 137.5.

Note now that, at $\hat{w}$, the shortest paths are not unique between the o-d pairs (a, b) and (a, c), since the paths $a - f - b$ and $a - i - c$ are also shortest. Furthermore, this set of weights can be improved by considering $P(\hat{w}, 2, 2, \ldots, 1)$, whose solution has every arc weight equal to 5 except that of arc ad, which is equal to 10 and where the objective function has the value 125. Moreover, examining every possible shortest path separately would not allow any progress because successively solving $P(\hat{w}, 2, 1, \ldots, 1)$ and $P(\hat{w}, 1, 2, \ldots, 1)$ still gives the same solution $\hat{w}$.

It is therefore crucial to consider every combination of shortest paths that are not unique at a potential solution.

4.5 Some properties of the algorithm

In this section, we examine some properties of the algorithm proposed above. In particular, we show its termination and analyse the "stability" of the local solution it produces.

Theorem 2 *The algorithm above terminates in a finite number of iterations.*

Proof. The number of paths between two nodes is finite, since the number of arcs m is finite. As a consequence, the number of different convex polygons $F(w_i, C_i)$ computed at Steps 1, and n_i^c calculated in Step 3 are also finite. The algorithm consists in a sequence of convex inverse shortest path problems differing by the actual shortest paths used in the explicit description of the constraints. Furthermore, each of these subproblems is considered at most once and is solvable in a finite number of operations. The complete algorithm therefore also terminates in a finite number of steps. □

Let us consider the point $\hat{w}$ obtained at termination of the algorithm. We now show that $\hat{w}$ is a local minimum of our problem (1)–(3) and analyse the neighbourhood $\mathcal{V}(\hat{w})$ around $\hat{w}$ in which every other feasible point has a higher objective function value. In other words, we show that $\hat{w}$ is locally "stable" as a local minimum in a neighbourhood of $\mathcal{V}(\hat{w})$ of $\hat{w}$ in which all the explicit shortest paths defining the constraints (3) remain unchanged when they are unique. The solution's "stability" therefore depends on "how far" the *second* shortest paths are from $\hat{w}$.

Considering the q-th shortest path constraint, we denote the cost of the "optimal" shortest path from o_q to d_q by P_q^1, that is

$$P_q^1 \stackrel{\text{def}}{=} \sum_{a \in p_q^1(\hat{w})} [\hat{w}]_a. \tag{17}$$

We already mentioned that $p_q^1(\hat{w})$ may be not unique, although P_q^1 is. We then define a *second* shortest path from o_q to d_q as a path whose cost is closest but strictly larger than that of $p_q^1(\hat{w})$, i.e. P_q^1. The first such second shortest path (in our predefined path order) is denoted, if it exists, by $p_q^2(\hat{w})$ and its cost by P_q^2. If $p_q^2(\hat{w})$ does not exist, then we set $P_q^2 = \infty$ by convention. With these additional notations, we are now in position to state the next property of our algorithm.

Theorem 3 *The point $\hat{w}$ computed by the algorithm is a local optimum of $P(w)$, the original problem. Moreover, $f(w) \geq f(\hat{w})$ for every w in*

$$\mathcal{V}(\hat{w}) \stackrel{\text{def}}{=} \left\{ w \in F \mid \|w - \hat{w}\|_1 < \min_q [P_q^2 - P_q^1] \right\}, \tag{18}$$

where $\|\cdot\|_1$ is the usual ℓ_1-norm.

Proof. Let us consider the conditions under which $p_q^1(\hat{w})$ may vary around $\hat{w}$, and define a stability neighbourhood $\mathcal{V}_q(\hat{w})$ associated with each shortest path constraint. Four cases need to be examined.

1. $P_q^2 = \infty$ and $n_q = 1$.
In this situation, the path from o_q to d_q is unique and $p_q^1(w)$ is obviously constant for all $w \in \mathbf{R}^m$. We then define $\mathcal{V}_q(\hat{w}) = \mathbf{R}^m \cap F = F$.

2. $P_q^2 = \infty$ and $n_q > 1$.
There are now more than one path from o_q to d_q, but they all have the same cost P_q^1. In this case, an infinitesimal change in the cost $\hat{w}$ may cause the feasible polygon defined at $\hat{w}$ to change. However, since $\hat{w}$ is a point produced by our algorithm, choosing any of the $n_q - 1$ other possible polygons does not produce an objective function decrease. This indicates that $f(\hat{w})$ may not be improved upon in the neighbourhood $\mathcal{V}_q(\hat{w}) = F$.

3. $P_q^2 \neq \infty$ and $n_q = 1$.
In this situation, the explicit description shortest path $p_q^1(\hat{w})$ will not change until its cost reaches that of the second shortest path. More precisely, $p_q^1(w)$ is constant in the neighbourhood

$$\mathcal{V}_q(\hat{w}) = \{w \in F : \|w - \hat{w}\|_1 < P_q^2 - P_q^1\}. \tag{19}$$

4. $P_q^2 \neq \infty$ and $n_q > 1$.

This is a combination of the two previous cases. As above, $f(w)$ cannot be improved upon in the neighbourhood $\mathcal{V}_q(\hat{w}) = \{w \in F : \|w - \hat{w}\|_1 < P_q^2 - P_q^1\}$.

Moreover, the algorithm's mechanism implies that we cannot find a point better than $\hat{w}$ by considering all combinations of constraint definitions as examined above for a single constraint. As a consequence, $\hat{w}$ will be a "stable" solution in the neighbourhood

$$\mathcal{V}(\hat{w}) = \bigcap_{q=1}^{n} \mathcal{V}_q(\hat{w}), = \left\{ w \in F : \|w - \hat{w}\|_1 < \min_q [P_q^2 - P_q^1] \right\}. \tag{20}$$

$\square$

We now examine the case where the original ISP problem also features lower bounds on the costs of the shortest paths between given origins and destinations, that is constraints of the type

$$0 \leq l_q \leq \sum_{a \in p_q^1(w)} [w]_a, \tag{21}$$

for $q = 1, \ldots, n$ (where l_q can be chosen as zero) and $l_q \leq u_q$. These constraints are much easier to handle, because the inequality (21) must be satisfied for *every* possible path from o_q to d_q. Of course, the number of these linear constraints is typically very high, but the situation is entirely similar to that handled in [4]. As a consequence, the technique developed in this last paper is directly applicable to each convex subproblem arising in the course of the solution of problem (1)–(3), (21).

5 Some numerical experiments

We now present some results obtained with a preliminary implementation of the algorithm described above. This implementation is in double precision Fortran 77 and has been run on a DECstation 3100 under Ultrix.

Our program selects among possible active shortest paths in Step 3 by examining first paths that differ as close to the destination as possible, ties being broken by considering nodes in their numbering order.

We have selected a few problems whose graphs, shortest path constraints and *a priori* weights have been generated in different ways. The problems and their characteristics are summarised in Table 1. In this table, the heading "nodes" refers to the number of nodes in the graph, "graph type" indicates how the network is generated, "weights" indicates how the *a priori* weights are chosen (a layered choice means that subset of arcs were chosen with constant costs, corresponding to grid levels in the case of grid-like graphs), "constraints" indicates how the shortest path constraints are chosen: either by choosing origins and destinations at random or by choosing them along the faces of the grids, when applicable.

All these problems but P1 were solved in that a local minimum was found for all of them.

The results of applying our pilot code to these problems is reported in Table 2. In this Table, i and $n^c = \sum_{j=1}^{i} n_j^c$ refer to the number of iterations of the algorithm and the total number

Problem	nodes	m	n	graph type	weights	constraints
Example	9	11	12	See Section 4.4		
P1	100	180	125	2D grid	constant	random + faces
P2	181	504	100	2D grid	constant	faces
P3	100	180	100	2D grid	layered	random
P4	100	210	100	random	random	random
P5	100	180	70	2D grid	layered	faces
P6	181	504	100	2D grid	layered	faces
P7	500	769	26	random	layered	random
P8	300	860	400	random	random	random

Table 1: Test examples and their characteristics

Problem	i	n^c	comb
Example	2	3	3
P1	-	$\approx 10^9$	-
P2	2	47	1
P3	1	0	0
P4	1	0	0
P5	1	0	0
P6	2	0	0
P7	1	0	0
P8	1	0	0

Table 2: Results for the test problems

of possible active shortest paths combinations respectively, as described in Section 4.4. The column "comb" indicates how many of the n^c paths combinations were effectively examined by the algorithm before termination. The symbol $-$ means that it has not been possible to solve problem P1 in less than a week on our workstation.

The following comments can be made on these results.

1. Many of our problems were solved with a single iteration ($i = 1$) of our algorithm, but not all of them. However, the number of iterations remains small on these examples.

2. As expected, problems with randomly generated weights were solved in a single iteration.

3. As shown be the behaviour of the algorithm on problem P1, the combinatorial aspect of the method may practically appear. The development of better heuristics to improve the choice of the active paths combination seems therefore useful.

4. The detection of paths with *equal* costs is nontrivial in finite precision. We have chosen to consider all paths whose relative costs differ by at most one hundred times machine precision. Further consideration should probably be given to this potentially important stability issue.

6 Conclusions and perspectives

We have presented and motivated the inverse shortest path problem with upper bounds on shortest paths costs. The NP-completeness of finding a global solution of this problem has then been shown. An algorithm for local minimisation has been presented, analysed and tested on a few examples.

Further research could cover heuristics for active path selection and stability analysis. We are also interested in applying the algorithm discussed in this paper to practical cases in traffic engineering.

7 Acknowledgements

The authors are indebted to Laurence Wolsey for interesting discussions on the inverse shortest paths problem. The financial support of the first author by the Belgian National Fund for Scientific Research is also gratefully acknowledged.

References

[1] R. Bellman, "On a routing problem", Quart. Appl. Math., vol. 16, pp. 87–90, 1958.

[2] D.P. Bertsekas, "An auction algorithm for shortest paths", SIAM Journal on Optimization (submitted), 1991.

[3] P.H.L. Bovy and E. Stern, "Route Choice: Wayfinding in Transport Networks", Kluwer Academic Publishers, Dordrecht, 1990.

[4] D. Burton and Ph. L. Toint, "On an instance of the inverse shortest paths problem", Mathematical Programming, vol. 53, pp. 45–61, 1992.

[5] D. Burton and Ph. L. Toint, "On the use of an inverse shortest paths algorithm for recovering linearly correlated costs", Mathematical Programming (submitted), 1992.

[6] S. Cook, "The complexity of Theorem Proving Procedures", Proc. 3rd Ann. ACM Symp. on Theory of Computing, Association for Computing Machinery, New York, 151–158, 1971.

[7] E.W. Dijkstra, "A note on two problems in connexion with graphs", Numerische Mathematik, vol. 1, pp. 269–271, 1959.

[8] S.M. Easa, “Shortest route with movement prohibition”, Transportation Research B, vol. 19, 3, pp. 197–208, 1985.

[9] L.R. Ford, “Network flow theory”, Report P-923, Rand Corp., Santa Monica, CA, 1956.

[10] D. Goldfarb and A. Idnani, “A Numerically Stable Dual Method for Solving Strictly Convex Quadratic Programs”, Mathematical Programming, vol. 27, pp. 1–33, 1983.

[11] M.R. Garey and D.S. Johnson, “Computers and intractability. A guide to the theory of NP-Completeness”, W.H. Freeman and Company, San Fransisco, 1979.

[12] D.B. Johnson, “Efficient algorithms for shortest paths in sparse networks”, J. Assoc. Comput. Mach., vol. 24, pp. 1–13, 1977.

[13] A. K. Louis and F. Natterer, “Mathematical problems of computerized tomography”, Proc. IEEE, vol. 71, no 3, pp. 379–389, 1983.

[14] P. Mirchandani and H. Soroush, “Generalized Traffic Equilibrium with Probabilistic Travel Times and Perceptions”, Transportation Science, vol. 21, no 3, pp. 133–152, 1987.

[15] E.F. Moore, “The shortest path through a maze”, in Proceedings of the International Symposium on the Theory of Switching, Part II, 1957, Harvard University, Cambridge, MA, pp. 295–292, 1959.

[16] T.J. Moser, “Shortest path calculation of seismic rays”, Geophysics, vol. 56, pp. 59–67, 1991.

[17] T.J. Moser, “The shortest path method for seismic ray tracing in complicated media”, Ph.D. Thesis, Rijksuniversiteit Utrecht, 1992.

[18] V.E. Outram and E. Thompson, “Driver's perceived cost in route choice”, Proceedings - PTRC Annual Meeting, London, pp. 226–257, 1978.