Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Inverse Optimization

Ravindra K. Ahuja, James B. Orlin,

To cite this article:
Ravindra K. Ahuja, James B. Orlin, (2001) Inverse Optimization. Operations Research 49(5):771-783. http://dx.doi.org/10.1287/opre.49.5.771.10607

INVERSE OPTIMIZATION

RAVINDRA K. AHUJA

Industrial and Systems Engineering Department, University of Florida, Gainesville, Florida 32611, ahuja@ufl.edu

JAMES B. ORLIN

Sloan School of Management, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, jorlin@mit.edu

(Received February 1998; revision received September 1999; accepted June 2000)

In this paper, we study inverse optimization problems defined as follows. Let **S** denote the set of feasible solutions of an optimization problem **P**, let c be a specified cost vector, and x^0 be a given feasible solution. The solution x^0 may or may not be an optimal solution of **P** with respect to the cost vector c. The inverse optimization problem is to perturb the cost vector c to d so that x^0 is an optimal solution of **P** with respect to d and $\|d-c\|_p$ is minimum, where $\|d-c\|_p$ is some selected L_p norm. In this paper, we consider the inverse linear programming problem under L_1 norm (where $\|d-c\|_p = \sum_{i\in J} w_j|d_j - c_j|$, with J denoting the index set of variables x_j and w_j denoting the weight of the variable j) and under L_∞ norm (where $\|d-c\|_p = \max_{j\in J}\{w_j|d_j-c_j|\}$). We prove the following results: (i) If the problem **P** is a linear programming problem, then its inverse problem under the L_1 as well as L_∞ norm is also a linear programming problem. (ii) If the problem **P** is a shortest path, assignment or minimum cut problem, then its inverse problem under the L_1 norm and unit weights can be solved by solving a problem of the same kind. For the nonunit weight case, the inverse problem reduces to solving a minimum cost flow problem. (iii) If the problem **P** is a minimum cost flow problem, then its inverse problem under the L_1 norm and unit weights reduces to solving a unit-capacity minimum cost flow problem. For the nonunit weight case, the inverse problem reduces to solving a minimum cost flow problem. (iv) If the problem **P** is a minimum cost flow problem, then its inverse problem under the L_∞ norm and unit weights reduces to solving a minimum mean cycle problem. For the nonunit weight case, the inverse problem reduces to solving a minimum cost-to-time ratio cycle problem. (v) If the problem **P** is polynomially solvable for linear cost functions, then inverse versions of **P** under the L_1 and L_∞ norms are also polynomially solvable.

1. INTRODUCTION

Inverse problems have been studied extensively by researchers working with geophysical data. Tarantola (1987) describes inverse problems in the following manner:

> *Let S represent a physical system. Assume that we are able to define a set of model parameters which completely describe S. All these parameters may not be directly measurable (such as the radius of Earth's metallic core). We can operationally define some observable parameters whose actual values hopefully depend on the values of the model parameters. To solve the forward problem is to predict the values of the observable parameters, given arbitrary values of the model parameters. To solve the inverse problem is to infer the values of the model parameters from given observed values of the observable parameters.*

In terms of the above notation, a typical optimization problem is a forward problem because it identifies the values of observable parameters (optimal decision variables), given the values of the model parameters (cost coefficients, right-hand side vector, and the constraint matrix). An inverse optimization problem consists of inferring the values of the model parameters (cost coefficients, right-hand side vector, and the constraint matrix), given the values of observable parameters (optimal decision variables). In the past few years, there has been an interest in inverse optimization problems in the operations research community, and a variety of inverse optimization problems have been studied by researchers.

In this paper, we study inverse optimization problems defined in the following manner. Let **P** denote an instance of an optimization problem with **S** as the set of feasible solutions and c as the cost vector; that is, $\mathbf{P} = \min\{\mathrm{c}x : x \in \mathbf{S}\}$. Suppose $x^0 \in \mathbf{S}$. The solution x^0 may or may not be an optimal solution of **P** with respect to the cost vector c. For a cost vector d, we define **P**(d) as the variation of problem **P** obtained by replacing the cost vector c replaced by d, that is, $\mathbf{P}(\mathrm{d}) = \min\{dx:\ x \in \mathbf{S}\}$. An inverse optimization problem with L_p norm is to identify a cost vector d such that x^0 is an optimal solution of **P**(d) and $\|\mathrm{d}-\mathrm{c}\|_p = [\sum_{j\in J}|d_j - c_j|^p]^{1/p}$ is minimum, where J denotes the index set of variables x_j. In words, the inverse optimization problem is to perturb the cost vector c to d so that x^0 is an optimal solution with respect to the perturbed cost vector and the cost of perturbation is minimum. In §2, we describe several applications of the inverse optimization problems and give references for some other applications.

Geophysical scientists were the first ones to study inverse problems. The book by Tarantola (1987) gives a comprehensive discussion of the theory of inverse problems in the geophysical sciences. Within the mathematical programming community, the interest in inverse optimization problems was generated by the papers by

Subject classifications: Programming, linear: inverse linear programming problems. Networks/graphs, flow algorithms: inverse shortest path, minimum cut, minimum cost flows. Transportation, mass transit: toll pricing in mass transportation networks.
Area of review: Optimization.

Burton and Toint (1992, 1994), who studied inverse shortest path problems arising in seismic tomography used in predicting the movement of earthquakes. In the past few years, inverse optimization problems have been studied rather intensively. In this paper, we give a unified presentation of several inverse problems. We first consider the inverse linear programming problem under the L_1 norm (where we minimize $\sum_{j\in J}\{w_j|d_j - c_j|\}$) and the L_∞ norm (where we minimize $\max_{j\in J}\{w_j|d_j - c_j|\}$). We then specialize these results for the following problems and obtain faster algorithms: the shortest path problem, the assignment problem, the minimum cut problem, and the minimum cost flow problem. Finally, we consider general inverse optimization problems under L_1 and L_∞ norms. We will henceforth refer to inverse problems under L_1 norm simply as inverse problems and inverse problems under L_∞ norm as minimax inverse problems. The major contributions made in this paper are as follows.

1. We show that if the problem **P** is a linear programming problem, then its inverse problems under the L_1 and L_∞ norms are also linear programming problems.

2. We show that if the problem **P** is a shortest path, assignment or minimum cut problem, then its inverse problem under the L_1 norm and unit weights can be solved by solving a problem of the same kind. For the nonunit weight case, the inverse problem reduces to solving a minimum cost flow problem.

3. We show that if the problem **P** is a minimum cost flow problem, then its inverse problem under the L_1 norm and unit weights reduces to solving a unit-capacity minimum cost flow problem. For the nonunit weight case, the inverse problem reduces to solving a minimum cost flow problem.

4. We show that if the problem **P** is a minimum cost flow problem, then its inverse problem under the L_∞ norm and unit weights reduces to solving a minimum mean cycle problem. For the nonunit weight case, the inverse problem reduces to solving a minimum cost-to-time ratio cycle problem. These results apply to the shortest path and assignment problems, too, which are special cases of the minimum cost flow problem.

5. We show that (under reasonable regularity conditions) if the problem **P** is polynomially solvable for linear cost functions, then inverse versions of **P** under the L_1 and L_∞ norms are also polynomially solvable. This result uses ideas from ellipsoid algorithm, and therefore does not lead to combinatorial algorithms for solving inverse optimization problems.

There has already been some research on inverse linear programming and inverse network flow problems under L_1 norm. Zang and Liu (1996) studied inverse assignment and minimum cost flow problems; Yang et al. (1997) and Zhang and Cai (1998) have studied the inverse minimum cut problems; and Xu and Zhang (1995) have studied the inverse shortest path problem. In this paper, we develop a unified framework from which algorithms for all inverse network flow problems are derived as special cases. Our algorithms either match the previous best algorithms or improve them, and at the same time obtain simpler proofs. We also study in this paper inverse linear programming and inverse network flow problems under L_∞ norm, which are new results.

In our other research papers we have studied the inverse spanning tree problem (Sokkalingam et al. 1999, Ahuja and Orlin 2000) and the inverse sorting problem (Ahuja and Orlin 2001). Ahuja and Orlin (1998) consider inverse network flow problems for the unit weight case and develop combinatorial proofs that do not rely on the inverse linear programming theory.

2. APPLICATIONS OF INVERSE OPTIMIZATION PROBLEMS

In this section, we describe briefly several applications of inverse optimization problems collected from the literature and provide references for a few other applications.

Geophysical Sciences and Medical Imaging

Geophysical scientists often do not have all the model parameters, because they may be very difficult or impossible to determine (such as the radius of Earth's metallic core). They may have some estimates of model parameters, and values of the observable parameters are used to improve the estimates of the model parameters. Consequently, inverse problems have been extensively studied by geophysical scientists (see, for example, Neumann-Denzau and Behrens 1984, Nolet 1987, Tarantola 1987, and Woodhouse and Dziewonski 1984). An important application in this area concerns predicting the movements of earthquakes. To model earthquake movements, consider a network obtained by the discretization of a geologic zone into a number of square cells. Nodes corresponding to adjacent cells are connected by arcs. The cost of an arc represents the transmission time of certain seismic waves from corresponding cells and is not accurately known. Earthquakes are then observed and the arrival times of the resulting seismic perturbations at various observation stations are observed. Assuming that the earthquakes travel along shortest paths, the problem faced by geologists is to reconstruct the transmission times between cells from the observation of shortest time waves and *a priori* knowledge of the geologic nature of the zone under study. This problem is an example of an inverse shortest path problem. Inverse problems also arise in X-ray tomography where observations from a CT-scan of a body part together with *a priori* knowledge of the body is used to estimate its dimension. The book by Tarantola (1987) gives a comprehensive treatment of the theory of inverse problems and provides additional applications.

Traffic Equilibrium

In a transportation network, users make a number of trips between different origin-destination pairs. Travel costs are flow dependent, and as the flow increases so does the travel costs. Drivers usually select their routes so as to minimize

their travel cost (or time). Under certain idealized assumptions, the resulting flow in such a network is a *user equilibrium flow*, where no user can decrease his or her travel cost unilaterally by changing his or her route (see, for example, Sheffi 1985). This user equilibrium flow does not necessarily correspond to the most efficient way of using the transportation network. A transportation planner may want to enforce a flow that minimizes the total travel cost over the network; such a flow is called the *system optimal flow*. A user equilibrium flow may or may not be the same as the system optimal flow. If not, then tolls may be imposed on some road segments of the route so that the user equilibrium flow becomes identical to the system optimal flow. If we denote by x^0 the system optimal flow, by x^* the user equilibrium flow, then imposing tolls amounts to changing travel costs so that the user equilibrium changes and becomes the same as the system optimal flow x^0. This is an example of the inverse optimization problem. In case the objective is to impose the minimum total toll to make the user equilibrium flow identical to the system optimal flow, then the resulting problem is an instance of the inverse optimization problem under the L_1 norm. In case the objective is to minimize the maximum toll imposed on any road, then the resulting problem is an instance of the inverse optimization problem under the L_∞ norm. As a matter of fact, these two problems are instances of the inverse multicommodity flow problem where flow between different origin-destination pairs is treated as a different commodity. This problem has been studied by Burton and Toint (1992, 1994) and Dial (1997). Burton and Toint studied nonlinear cost inverse path problems, and Dial studied linear-cost, single-source shortest path problems. Our results in this paper generalize Dial's results.

The linear impedance problem arising in the railroad scheduling is also an instance of the inverse multicommodity flow problem where a cost structure is required which would make a specified "target flow" optimal in the railroad network (Shan 1999). Inverse problems also find applications in portfolio optimization (Carr and Lovejoy 1997 and Dembo et al. 1998). It also has some plausible applications in isotonic regression (Ahuja and Orlin 2001) and stability analysis (Greenberg 1997).

3. FORMULATING THE INVERSE LINEAR PROGRAMMING PROBLEM

We consider the inverse version of the following linear programming problem, which we shall subsequently refer to as **LP**:

$$\text{Minimize} \sum_{j\in J} c_j x_j, \tag{1a}$$

subject to

$$\sum_{j\in J} a_{ij} x_j \geqslant b_i \quad \text{for all } i \in I, \tag{1b}$$

$$x_j \geqslant l_j \quad \text{for all } j \in J, \tag{1c}$$

$$-x_j \geqslant -u_j \quad \text{for all } j \in J, \tag{1d}$$

where J denotes the index set of the decision vector x and I denotes the index set of the constraints, l_j denotes the lower bound on x_j, and u_j denotes the upper bound on x_j. Let us associate the dual variable π_i with the ith constraint in Equation (1b), λ_j with the jth constraint in (1c), and φ_j with the jth constraint in (1d). The dual of LP is the following linear program:

$$\text{Maximize} \sum_{i\in I} b_i \pi_i + \sum_{j\in J} l_j \lambda_j - \sum_{j\in J} u_j \varphi_j, \tag{2a}$$

subject to

$$\sum_{i\in I} a_{ij} \pi_i + \lambda_j - \varphi_j = c_j \quad \text{for all } j \in J, \tag{2b}$$

$$\pi_i \geqslant 0 \quad \text{for all } i \in I; \quad \lambda_j \geqslant 0 \quad \text{for all } j \in J; \quad \text{and} \quad \varphi_j \geqslant 0 \quad \text{for all } j \in J. \tag{2c}$$

One form of the linear programming optimality conditions states that the primal solutions x and the dual solution (π, λ, φ) are optimal for their respective problems if x is feasible for Equations (1), (π, λ, φ) is feasible for Equations (2), and together they satisfy the following complementary slackness conditions:

$$\text{(i)} \quad \text{if } \sum_{j\in J} a_{ij} x_j > b_i \text{ then } \pi_i = 0, \tag{3a}$$

$$\text{(ii)} \quad \text{if } x_j > l_j \text{ then } \lambda_j = 0, \tag{3b}$$

$$\text{(iii)} \quad \text{if } x_j < u_j \text{ then } \varphi_j = 0. \tag{3c}$$

Let x^0 be a feasible solution of Equations (1). Let B denote the index set of binding constraints in (1b) with respect to x^0 (that is, $B = \{i \in I\colon \sum_{j\in J} a_{ij} x_j = b_i\}$), L denote the index set of binding constraints in Equation (1c) (that is, $L = \{j \in J\colon x_j^0 = l_j\}$), and U denote the index set of binding constraints in Equation (1d) (that is, $U = \{j \in J\colon x_j^0 = u_j\}$). Let $F = \{j \in J\colon 0 < x_j^0 < u_j\}$. We can restate the complementary slackness conditions (3) as

$$\text{(i)} \quad \pi_i = 0 \quad \text{for all } i \notin B, \tag{4a}$$

$$\text{(ii)} \quad \lambda_j = 0 \quad \text{for all } j \notin L, \tag{4b}$$

$$\text{(iii)} \quad \varphi_j = 0 \quad \text{for all } j \notin U. \tag{4c}$$

We want to make x^0 an optimal solution of Equations (1) by perturbing the cost vector c to d. We denote by **LP**(d) the linear program (1) where the c_js have been replaced with d_js. We call d *inverse feasible* with respect to x^0 if x^0 is an optimal solution of **LP**(d). Now notice that x^0 is an optimal solution of LP(d) if and only if there exists a dual solution (π, λ, φ) that satisfies (2b) with c_j replaced by d_j and the primal-dual pair satisfies the complementary slackness conditions (4). Using conditions (4) in Equation (2) gives us the following characterization of inverse feasible

cost vectors:

$$\sum_{i\in B} a_{ij}\pi_i + \lambda_j = d_j \quad \text{for all } j \in L, \tag{5a}$$

$$\sum_{i\in B} a_{ij}\pi_i - \varphi_j = d_j \quad \text{for all } j \in U, \tag{5b}$$

$$\sum_{i\in B} a_{ij}\pi_i = d_j \quad \text{for all } j \in F, \tag{5c}$$

$$\pi_i \geqslant 0 \quad \text{for all } i \in B; \quad \lambda_j \geqslant 0 \quad \text{for all } j \in L; \quad \text{and} \quad \varphi_j \geqslant 0 \quad \text{for all } j \in U. \tag{5d}$$

The inverse problem is to minimize $\|\mathrm{d} - \mathrm{c}\|_p$ over all inverse feasible cost vectors d (that is, all cost vectors d satisfying Equations (5)). In the next two sections, we will consider the following two objective functions for the inverse problem: (i) minimize $\sum_{j\in J} w_j|\mathrm{d}_j - \mathrm{c}_j|$, and (ii) minimize $\max\{w_j|\mathrm{d}_j - \mathrm{c}_j| : j \in J\}$.

4. SOLVING THE INVERSE LINEAR PROGRAMMING PROBLEM UNDER THE L_1 NORM

In this section, we will consider the inverse linear programming problem under the weighted L_1 norm. A similar approach was taken by Zhang and Liu (1996). They showed that the inverse problem under the L_1 norm was solvable as a linear program. We first extend their results to the weighted case. More importantly, we clarify the connection between the dual of the inverse problem and a relaxation of the original problem. Zhang and Liu observed this connection for the assignment problem, and here we establish that the connection extends to all linear programs. We also show that in some network flow problems, the solution to the inverse problem is readily obtainable from solving the original problem and using reduced cost information appropriately. In particular, we show that when applied to 0-1 integer linear programming problems, the inverse problem takes on a particular elegant form.

In this section, we will consider the inverse linear programming problem under the weighted L_1 norm. We have shown in the previous section that this reduces to solving minimize $\sum_{j\in J} w_j|\mathrm{d}_j - \mathrm{c}_j|$, subject to Equations (5). This is not a linear programming problem in its current form but can be easily converted into one using a standard transformation. It is well known that minimizing $w_j|\mathrm{d}_j - \mathrm{c}_j|$ is equivalent to minimizing $\alpha_j + \beta_j$, subject to $\mathrm{d}_j - \mathrm{c}_j = \alpha_j - \beta_j$, $\alpha_j \geqslant 0$ and $\beta_j \geqslant 0$. Using this transformation, the inverse linear programming problem can be stated as follows:

$$\text{Minimize} \ \sum_{j\in J} w_j\alpha_j + \sum_{j\in J} w_j\beta_j,$$

or, equivalently,

$$\text{maximize} \ -\sum_{j\in J} w_j\alpha_j - \sum_{j\in J} w_j\beta_j, \tag{6a}$$

subject to

$$\sum_{i\in B} a_{ij}\pi_i - \alpha_j + \beta_j + \lambda_j = c_j \quad \text{for all } j \in L, \tag{6b}$$

$$\sum_{i\in B} a_{ij}\pi_i - \alpha_j + \beta_j - \varphi_j = c_j \quad \text{for all } j \in U, \tag{6c}$$

$$\sum_{i\in B} a_{ij}\pi_i - \alpha_j + \beta_j = c_j \quad \text{for all } j \in F, \tag{6d}$$

$$\pi_i \geqslant 0 \quad \text{for all } i \in B; \quad \alpha_j \geqslant 0 \quad \text{and} \quad \beta_j \geqslant 0 \quad \text{for all } j \in J, \tag{6e}$$

$$\lambda_j \geqslant 0 \quad \text{for all } j \in L; \quad \text{and} \quad \varphi_j \geqslant 0 \quad \text{for all } j \in U. \tag{6f}$$

We will now simplify Equations (6). We first note that in an optimal solution of (6), both α_j and β_j cannot take positive values because otherwise we can reduce both by a small amount δ without violating any constraint and strictly improving the objective function value. We can restate Equations (6b), (6c), and (6d) as

$$-\alpha_j + \beta_j = c_j^\pi - \lambda_j \quad \text{for all } j \in L, \tag{7a}$$

$$-\alpha_j + \beta_j = c_j^\pi + \varphi_j \quad \text{for all } j \in U, \tag{7b}$$

$$-\alpha_j + \beta_j = c_j^\pi \quad \text{for all } j \in F, \tag{7c}$$

where $c_j^\pi = c_j - \sum_{i\in B} a_{ij}\pi_i$. There are three cases to consider.

Case 1. $c_j^\pi > 0$. The nonnegativity of α_j and β_j and the fact that we wish to minimize $\alpha_j + \beta_j$ implies that (i) if $j \in L$, then $\lambda_j = c_j^\pi = |c_j^\pi|$, $\alpha_j = \beta_j = 0$ and hence $d_j = c_j$; and (ii) if $j \in F \cup U$ then $\alpha_j = \varphi_j = 0$, $\beta_j = c_j^\pi = |c_j^\pi|$, and hence $d_j = c_j - |c_j^\pi|$.

Case 2. $c_j^\pi < 0$. In this case, (i) if $j \in U$ then $\varphi_j = -c_j^\pi = |c_j^\pi|$, $\alpha_j = \beta_j = 0$, hence $d_j = c_j$; and (ii) if $j \in F \cup L$, then $\beta_j = \lambda_j = 0$, $\alpha_j = -c_j^\pi = |c_j^\pi|$, and hence $d_j = c_j + |c_j^\pi|$.

Case 3. $c_j^\pi = 0$. In this case, $\alpha_j = \beta_j = \lambda_j = \varphi_j = 0$, and hence $d_j = c_j$.

The preceding case analysis implies that if π denotes the optimal solution of Equations (6), then the optimal cost vector d^* is given by

$$d_j^* = \begin{cases} c_j - |c_j^\pi| & \text{if } c_j^\pi > 0 \text{ and } x_j^0 > 0, \\ c_j + |c_j^\pi| & \text{if } c_j^\pi < 0 \text{ and } x_j^0 < u_{ij}, \\ c_j & \text{otherwise.} \end{cases} \tag{8}$$

We have shown above that we can solve the inverse problem by solving Equations (6), and the optimal values of π can be used to obtain the optimal cost vector using Equation (8). Instead of solving (6) we can alternatively solve the dual of (6), which turns out to be a variation of the original problem Equations (1). We associate the variable y_j with the constraint associated with the jth index in Equations (6b)–(6d) and then take its dual. We get the

following linear programming problem:

$$\text{Minimize } \sum_{j\in J} c_j y_j, \tag{9a}$$

subject to

$$\sum_{j\in J} a_{ij} y_j \geqslant 0 \quad \text{for all } i \in B, \tag{9b}$$

$$0 \leqslant y_j \leqslant w_j \quad \text{for all } j \in L, \tag{9c}$$

$$-w_j \leqslant y_j \leqslant 0 \quad \text{for all } j \in U, \tag{9d}$$

$$-w_j \leqslant y_j \leqslant w_j \quad \text{for all } j \in F. \tag{9e}$$

We can formulate the dual inverse problem in an alternate manner that may be more convenient to work with compared to the formulation in Equations (9). Substituting $y_j = x_j - x_j^0$ for each $j \in J$ gives us the following equivalent formulation that is more similar to the original formulation of **LP**:

$$\text{Minimize } \sum_{j\in J} c_j x_j, \tag{10a}$$

subject to

$$\sum_{j\in J} a_{ij} x_j \geqslant b_i \quad \text{for all } i \in B, \tag{10b}$$

$$0 \leqslant x_j \leqslant w_j \quad \text{for all } j \in L, \tag{10c}$$

$$u_j - w_j \leqslant x_j \leqslant u_j \quad \text{for all } j \in U, \tag{10d}$$

$$x_j^0 - w_j \leqslant x_j \leqslant x_j^0 + w_j \quad \text{for all } j \in F. \tag{10e}$$

The formulations (9) and (10) of the dual of the inverse linear programming problem are equivalent to one another. The two formulations have different primal optimal solutions and are related using the formula $x = x^0 - y$. But they have the same optimal dual solution π from which we may determine the optimal cost vector d^*. We refer to the formulation (9) as *the 0-centered dual inverse problem*, and to the formulation (10) as *the x^0-centered dual inverse problem*.

In the formulation of the problem **LP**, we have assumed that all inequalities are of the form "$\geqslant$". In case we have some "$\leqslant$" inequalities in Equations (1), then in the 0-centered problem or the x^0-centered problem the corresponding constraint (if binding) will also be a "$\leqslant$" inequality. In case we have an equality constraint in Equations (1), then this constraint will be a binding constraint, and the corresponding constraint will always be present in the 0-centered dual inverse problem or the x^0-centered dual inverse problem.

5. THE 0-1 LINEAR PROGRAMMING PROBLEM WITH UNIT WEIGHTS AND UNDER L_1 NORM

We will now consider a special case of the bounded variable linear programming problem in which each lower bound equals 0, each upper bound equals one, and there always exists an integer optimal solution. We refer to such a linear programming problem as *a 0-1 linear programming problem*. We also assume that $w_j = 1$ for each $j \in J$. Several combinatorial optimization problems, such as the single-source single-sink shortest path problem, the assignment problem, and the minimum cut problem, can be formulated as 0-1 linear programming problems. Let x^0 be a 0-1 feasible solution of a 0-1 linear programming problem we wish to make optimal by perturbing the cost vector c to d. Let B denote the index set of constraints binding with respect to the solution x^0. Because x^0 is a 0-1 solution, each index $j \in J$ either belongs to L or U, and in both the cases (10c) and (10d) reduce to $0 \leqslant x_j \leqslant 1$. We thus get the following x^0-centered dual inverse problem:

$$\text{Minimize } \sum_{j\in J} c_j x_j, \tag{11a}$$

subject to

$$\sum_{j\in J} a_{ij} x_j \geqslant b_i \quad \text{for all } i \in B, \tag{11b}$$

$$0 \leqslant x_j \leqslant 1 \quad \text{for all } j \in J, \tag{11c}$$

which is the same as the original problem except that the nonbinding constraints with respect to x^0 have been eliminated. In the case when all constraints are binding (for example, when each constraint in Equation (11b) is an equality constraint), $B = I$ and its x^0-centered dual inverse problem is the same as the original problem.

In the case of the 0-1 linear programming problem, we can restate the expression for computing the optimal cost vector d^*. Let x^* be an optimal solution of Equations (11) and π denote the optimal dual variables associated with the constraints in (11b). It follows from the linear programming theory that (i) $c_j^\pi < 0$ if and only if $x_j^* = u_j$, and (ii) $c_j^\pi > 0$ if and only if $x_j^* = 0$. Using these results in Equations (10) yields the following optimal cost vector:

$$d_j^* = \begin{cases} c_j - |c_j^\pi| & \text{for all } j \text{ satisfying } x_j^0 = 1 \text{ and } x_j^* = 0, \\ c_j + |c_j^\pi| & \text{for all } j \text{ satisfying } x_j^0 = 0 \text{ and } x_j^* = 1, \\ c_j & \text{for all } j \text{ satisfying } x_j^0 = x_j^*. \end{cases} \tag{12}$$

In this case the optimal objective function value for the inverse problem is $\sum_{\{j\in J:\, x_j^0 \neq x_j^*\}} |c_j^\pi|$.

6. SOLVING THE INVERSE LINEAR PROGRAMMING PROBLEM UNDER THE L_∞ NORM

In this section, we study the inverse of the linear programming problem **LP** under the L_∞ norm, called the *minimax inverse linear programming problem*. In this problem, we wish to obtain an inverse feasible cost vector d that minimizes $\max\{w_j|d_j - c_j|:\ j \in J\}$, where $w_j \geqslant 0$

for all $j \in J$. It follows from the discussion in §3 that the minimax inverse linear programming problem is to minimize $\max\{w_j|d_j - c_j|: j \in J\}$, subject to Equations (5). This mathematical program is not a linear program because it contains absolute signs on terms in the objective function and a maximization of terms instead of summation of terms; however, it can be converted to a linear programming problem by using well known transformations. To eliminate the absolute signs in the objective function, we replace $w_j|d_j - c_j|$ by $w_j\alpha_j + w_j\beta_j$, subject to $d_j - c_j = \alpha_j - \beta_j$, $\alpha_j \geqslant 0$ and $\beta_j \geqslant 0$. Further, to eliminate the maximization of the terms, we introduce a nonnegative variable θ, and add the constraints $w_j\alpha_j + w_j\beta_j \leqslant \theta$ for each $j \in J$ to ensure that each term is less than or equal to θ. We also convert the minimization form of the objective function into the maximization form. This gives us the following linear programming problem:

$$\text{Maximize } -\theta, \tag{13a}$$

subject to

$$\sum_{i\in B} a_{ij}\pi_i - \alpha_j + \beta_j + \lambda_j = c_j \quad \text{for all } j \in L, \tag{13b}$$

$$\sum_{i\in B} a_{ij}\pi_i - \alpha_j + \beta_j - \varphi_j = c_j \quad \text{for all } j \in U, \tag{13c}$$

$$\sum_{i\in B} a_{ij}\pi_i - \alpha_j + \beta_j = c_j \quad \text{for all } j \in F, \tag{13d}$$

$$w_j\alpha_j + w_j\beta_j - \theta \leqslant 0 \quad \text{for all } j \in J, \tag{13e}$$

$$\pi_i \geqslant 0 \quad \text{for all } i \in B; \quad \alpha_j \geqslant 0 \quad \text{and}$$
$$\beta_j \geqslant 0 \quad \text{for all } j \in J, \tag{13f}$$

$$\lambda_j \geqslant 0 \quad \text{for all } j \in L; \quad \text{and}$$
$$\varphi_j \geqslant 0 \quad \text{for all } j \in U. \tag{13g}$$

Let $c_j^\pi = c_j - \sum_{i\in B} a_{ij}\pi_i$. We can restate Equations (13b), (13c), and (13d) as

$$-\alpha_j + \beta_j = c_j^\pi - \lambda_j \quad \text{for all } j \in L, \tag{14a}$$

$$-\alpha_j + \beta_j = c_j^\pi + \varphi_j \quad \text{for all } j \in U, \tag{14b}$$

$$-\alpha_j + \beta_j = c_j^\pi \quad \text{for all } j \in F, \tag{14c}$$

There are three cases to consider.

Case 1. $c_j^\pi > 0$. The nonnegativity of α_j and β_j and the fact that we wish to minimize the maximum of $w_j\alpha_j + w_j\beta_j$ implies that there always exist an optimal solution such that (i) if $j \in L$ then $\lambda_j = c_j^\pi = |c_j^\pi|, \alpha_j = \beta_j = 0$, and hence $d_j = c_j$; and (ii) if $j \in F \cup U$ then $\alpha_j = \varphi_j = 0, \beta_j = c_j^\pi = |c_j^\pi|$, and hence, $d_j = c_j - |c_j^\pi|$. To summarize, in case $c_j^\pi > 0$, the constraints (13b)–(13e) reduce to $w_j c_j^\pi \leqslant \theta$ for all $j \in F \cup U$.

Case 2. $c_j^\pi < 0$. In this case, (i) if $j \in U$ then $\varphi_j = -c_j^\pi = |c_j^\pi|, \alpha_j = \beta_j = 0$ and hence $d_j = c_j$; and (ii) if $j \in F \cup L$ then $\beta_j = \lambda_j = 0$ and $\alpha_j = -c_j^\pi = |c_j^\pi|$, and hence $d_j = c_j + |c_j^\pi|$. To summarize, in case $c_j^\pi < 0$, the constraints (13b)–(13e) reduce to $-w_j c_j^\pi \leqslant \theta$ for all $j \in F \cup L$.

Case 3. $c_j^\pi = 0$. In this case, $\alpha_j = \beta_j = \lambda_j = \varphi_j = 0$, and $d_j = c_j$. In this case, the Constraint (6.1e) is vacuously satisfied. To summarize, in case $c_j^\pi = 0$, the constraints (13b)–(13e) remain satisfied.

The previous case analysis also implies that the optimal cost vector $\mathrm{d}_j^* = \mathrm{c}_j + \alpha_j - \beta_j$ is given by

$$\mathrm{d}_j^* = \begin{cases} \mathrm{c}_j - |\mathrm{c}_j^\pi| & \text{if } \mathrm{c}_j^\pi > 0 \text{ and } x_j^0 > 0, \\ \mathrm{c}_j + |\mathrm{c}_j^\pi| & \text{if } \mathrm{c}_j^\pi < 0 \text{ and } x_j^0 < u_{ij}, \\ c_j & \text{otherwise,} \end{cases} \tag{15}$$

which is the same as in the case of L_1 norm. The preceding analysis also allows us to formulate Equations (14) as in the following linear program:

$$\text{Maximize } -\theta, \tag{16a}$$

subject to

$$-w_j\Big(c_j - \sum_{i\in B} a_{ij}\pi_i\Big) \leqslant \theta, \quad \text{for all } j \in F \cup L, \tag{16b}$$

$$w_j\Big(c_j - \sum_{i\in B} a_{ij}\pi_i\Big) \leqslant \theta, \quad \text{for all } j \in F \cup U, \tag{16c}$$

$$\pi_i \geqslant 0 \quad \text{for all } i \in B, \tag{16d}$$

which can be reformulated as

$$\text{Maximize } -\theta, \tag{17a}$$

subject to

$$\sum_{i\in B} a_{ij}\pi_i - \frac{1}{w_j}\theta \leqslant c_j \quad \text{for all } j \in F \cup L, \tag{17b}$$

$$-\sum_{i\in B} a_{ij}\pi_i - \frac{1}{w_j}\theta \leqslant -c_j \quad \text{for all } j \in F \cup U, \tag{17c}$$

$$\pi_i \geqslant 0 \quad \text{for all } i \in B, \tag{17d}$$

where for simplicity of exposition we assume that $w_j \neq 0$ for each $j \in J$. By taking the dual of Equations (17), we can obtain an equivalent formulation of the minimax inverse problem. We associate the variable y_j^+ with the constraint (17b) and the variable y_j^- with the constraint (17c). The dual of Equations (17) is the following linear programming problem:

$$\text{Minimize } \sum_{j\in L} c_j y_j^+ + \sum_{j\in F} c_j(y_j^+ - y_j^-) - \sum_{j\in U} c_j y_j^-, \tag{18a}$$

subject to

$$\sum_{j\in L} a_{ij} y_j^+ + \sum_{j\in F} a_{ij}(y_j^+ - y_j^-) - \sum_{j\in U} a_{ij} y_j^- \geqslant 0 \quad \text{for all } i \in B, \tag{18b}$$

$$\sum_{j\in L} \frac{1}{w_j} y_j^+ + \sum_{j\in F} \frac{1}{w_j}(y_j^+ + y_j^-) + \sum_{j\in U} \frac{1}{w_j} y_j^- = 1, \tag{18c}$$

$$y_j^+ \geqslant 0, y_j^- \geqslant 0 \quad \text{for all } j \in J. \tag{18d}$$

Let $y_j = y_j^+$ for all $j \in L$, $y_j = (y_j^+ - y_j^-)$ for all $j \in F$, and $y_j = -y_j^-$ for all $j \in U$. In terms of the variables y_js, we can reformulate Equations (18) as follows:

$$\text{Minimize} \sum_{j \in J} c_j y_j, \tag{19a}$$

subject to

$$\sum_{j \in J} a_{ij} y_j \geqslant 0 \quad \text{for all } i \in B, \tag{19b}$$

$$\sum_{j \in J} \frac{1}{w_j} |y_j| = 1, \tag{19c}$$

$$y_j \geqslant 0 \quad \text{for all} \quad j \in L, \quad \text{and} \quad y_j \leqslant 0 \quad \text{for all } j \in U. \tag{19d}$$

We have assumed in Equations (19) that $w_j \neq 0$ for each $j \in J$. It is easy to see that in the case when some w_j have zero values, the formulation will be identical to that in (19), except that J is replaced by J' where $J' = \{j \in J: w_j \neq 0\}$.

If π denotes the optimal dual variables associated with (19b), then the optimal cost vector d^* can be computed using Equations (16). We refer to the formulation (19) as *the 0-centered minimax dual inverse problem*. We can obtain the *x^0-centered minimax dual inverse problem* by substituting $y_j = x_j - x_j^0$ for each $j \in J$ in (19). This gives us the following equivalent formulation of the minimax inverse problem:

$$\text{Minimize} \sum_{j \in J} c_j x_j, \tag{20a}$$

subject to

$$\sum_{j \in J} a_{ij} x_j \geqslant b_i \quad \text{for all } i \in B, \tag{20b}$$

$$\sum_{j \in J} \frac{1}{w_j} |x_j - x_j^0| \leqslant 1, \tag{20c}$$

$$x_j \geqslant x_j^0 \quad \text{for all } j \in L, \quad \text{and} \quad x_j \leqslant x_j^0 \quad \text{for all } j \in U. \tag{20d}$$

In this section we have assumed so far that all inequalities are of the form "$\geqslant$". In case we have some "$\leqslant$" inequalities, then in the 0-centered problem or the x^0-centered problem, the corresponding constraint (if binding) will also be a "$\leqslant$" inequality. In case we have an equality constraint, then this constraint will always be a binding constraint and the corresponding constraint will always be present in the 0-centered or the x^0-centered minimax dual inverse problem.

7. THE INVERSE SHORTEST PATH PROBLEM UNDER L_1 NORM

In this section, we study the inverse version of the single-source, single-sink, shortest path problem. We formulate the single-source, single-sink problem as a 0-1 integer program and apply the results of §5 to show that the inverse shortest path problem reduces to solving a shortest path problem. This problem has also been studied by Zhang and Liu (1996), who suggest solving the problem by first transforming to an inverse assignment problem and then solving the inverse assignment problem by solving an instance of the assignment problem. Our approach requires solving a shortest path problem, which is a more efficient method to solve compared to an assignment problem.

Let $G = (N, A)$ be a directed network, where N denotes the node set and A denotes the arc set. Let nodes s and t denote two specified nodes. Let us associate a cost c_{ij} for each arc $(i, j) \in A$. The *s-t* shortest path problem is to determine a directed path from node s to node t in G (henceforth called an *s-t path*) whose cost, given by $\sum_{(i,j) \in P} c_{ij}$, is minimum among all *s-t* paths in G. The shortest path problem can be formulated as the following linear programming problem:

$$\text{Minimize} \sum_{(i,j) \in A} c_{ij} x_{ij}, \tag{21a}$$

subject to

$$\sum_{\{j:(i,j) \in A\}} x_{ij} - \sum_{\{j:(j,i) \in A\}} x_{ji} = \begin{cases} 1 & \text{for } i = s, \\ 0 & \text{for all } i \notin \{s, t\}, \\ -1 & \text{for } i = t, \end{cases} \tag{21b}$$

$$0 \leqslant x_{ij} \leqslant 1 \quad \text{for all } (i, j) \in A. \tag{21c}$$

We assume that the network G does not contain any negative cost cycle; under this assumption Equations (21) are valid formulation of the shortest path problem. Moreover, it has a 0-1 optimal solution. In the inverse shortest path problem, we are given an *s-t* path $\mathbf{P}^0$ in G that we wish to make a shortest *s-t* path by perturbing the arc costs. Let x^0 be the flow corresponding to $\mathbf{P}^0$, that is, $x_{ij}^0 = 1$ for all $(i, j) \in \mathbf{P}^0$ and $x_{ij}^0 = 0$ for all $(i, j) \notin \mathbf{P}^0$. The shortest path problem is a special case of the 0-1 linear programming problem, and it follows from our discussion in §5 that the unit-weight x^0-centered dual inverse shortest path problem is identical to Equations (21) because all constraints in (21b) are binding constraints. Let $\mathbf{P}^*$ denote the shortest *s-t* path in G with c_{ij} as arc costs and let x^* denote the corresponding 0-1 flow. For the shortest path problem, the reduced cost of an arc (i, j) is given by $c_{ij}^\pi = c_{ij} - \pi_i + \pi_j$. It is well known (see, for example, Ahuja et al. 1993) that reduced costs corresponding to the shortest path $\mathbf{P}^*$ satisfy the following conditions:

$$c_{ij}^\pi = 0 \quad \text{for all } (i, j) \in \mathbf{P}^*, \tag{22a}$$

$$c_{ij}^\pi \geqslant 0 \quad \text{for all } (i, j) \notin \mathbf{P}^*. \tag{22b}$$

Let $\mathbf{P}^* \backslash \mathbf{P}^0 = \{(i, j) \in A: (i, j) \in \mathbf{P}^* \text{ and } (i, j) \notin \mathbf{P}^0\}$, and $\mathbf{P}^0 \backslash \mathbf{P}^* = \{(i, j) \in A: (i, j) \in \mathbf{P}^0 \text{ and } (i, j) \notin \mathbf{P}^*\}$. Using these results in Equations (12) gives the following optimal cost vector d^*:

$$d_{ij}^* = \begin{cases} c_{ij} - c_{ij}^\pi & \text{for all } (i, j) \in \mathbf{P}^0 \backslash \mathbf{P}^*, \\ c_{ij} & \text{for all } (i, j) \notin \mathbf{P}^0 \backslash \mathbf{P}^*. \end{cases} \tag{23}$$

In words, the above result implies that for each arc that is in $\mathbf{P}^0$ but not in $\mathbf{P}^*$, we decrease the arc cost by an amount equal to the optimal reduced cost of the arc. The cost of every other arc remains unchanged. This change decreases the cost of the path $\mathbf{P}^0$ by $\sum_{(i,j)\in \mathbf{P}^0\setminus\mathbf{P}^*} c_{ij}^{\pi}$ units and does not affect the cost of the path $\mathbf{P}^*$. After this change, the modified reduced cost of each arc (i, j) in $\mathbf{P}^0$ becomes 0, and it becomes an alternate shortest s-t path in G.

We have shown above that the unit-weight inverse shortest path problem can be solved by solving a shortest path problem. When all arc costs are nonnegative, we can solve the shortest path problem in $O(m + n\log n)$ time using Fredman and Tarjan's (1984) implementation of Dijkstra's algorithm. In case some arc costs are negative, we can solve the shortest path problem in $O(nm)$ time using the FIFO label correcting algorithm (see, for example, Ahuja et al. 1993), or in $\mathbf{O}(\sqrt{n}\, m \log \mathbf{C})$ time using Goldberg's (1995) algorithm, where $\mathbf{C} = \max\{|c_{ij}|: (i, j) \in A\}$.

In the weighted version of the inverse shortest path problem, the resulting x^0-centered dual inverse problem is a minimum cost flow problem and can be solved using any efficient minimum cost flow algorithm.

We note that one could also address the single-source, multiple-sink problem using the results of §5, but only if one were to model the problem as a 0-1 integer programming formulation. The usual way of modeling the single-source, multiple-sink problem is as a single commodity flow, which is not a 0-1 integer program. However, an alternative formulation is as a multicommodity flow problem, which is a 0-1 integer program and can use the results of §5.

8. THE INVERSE ASSIGNMENT PROBLEM UNDER L_1 NORM

In this section, we study the inverse version of the assignment problem. The solution for the inverse assignment problem is readily obtained from the solution to the same assignment problem. This result was first established by Zhang and Liu [1996]. Here we show that the result follows directly from the results of §6.

Let $G = (N_1 \cup N_2, A)$ be a bipartite directed network with $|N_1| = |N_2|$, and $A \subseteq N_1 \times N_2$. We associate a cost c_{ij} for each arc $(i, j) \in A$. The assignment problem is the following linear programming problem:

$$\text{Minimize} \sum_{(i,j)\in A} c_{ij}x_{ij}, \tag{24a}$$

subject to

$$\sum_{\{j:(i,j)\in A\}} x_{ij} = 1 \quad \text{for all } i \in N_1, \tag{24b}$$

$$-\sum_{\{i:(i,j)\in A\}} x_{ij} = -1 \quad \text{for all } i \in N_2, \tag{24c}$$

$$0 \leqslant x_{ij} \leqslant 1 \quad \text{for all } (i, j) \in A. \tag{24d}$$

Each 0-1 solution x of Equations (24) defines an assignment $M = \{(i, j) \in A: x_{ij} = 1\}$. Conversely, each assignment M defines a solution x of (24). In the inverse assignment problem, we are given an assignment M^0 in G, which we wish to make optimal by perturbing the arc costs. As in the case of the shortest path problem, the assignment problem is a special case of the 0-1 linear programming problem (11), and its x^0-centered dual inverse problem for the unit-weight case is the same as (24). Let M^* denote the optimal assignment in G and let $c_{ij}^{\pi} = c_{ij} - \pi_i + \pi_j$ denote the optimal reduced costs of arcs. The optimal reduced costs satisfy the condition that $c_{ij}^{\pi} = 0$ for all $(i, j) \in M^*$, and $c_{ij}^{\pi} \geqslant 0$ for all $(i, j) \notin M^*$. Using this result in Equation (12) gives us the following optimal cost vector d^* for the inverse assignment problem:

$$d_{ij}^* = \begin{cases} c_{ij} - c_{ij}^{\pi} & \text{for all } (i, j) \in M^0\setminus M^*, \\ c_{ij} & \text{for all } (i, j) \notin M^0\setminus M^*. \end{cases} \tag{25}$$

Currently, the best available strongly polynomial time bound to solve the assignment problem is $\mathbf{O}(nm + n^2 \log n)$ and is attained by several algorithms (see, for example, Goldfarb 1985). The best available weakly polynomial algorithm is due to Gabow and Tarjan (1989) and Orlin and Ahuja (1992), and it runs in $\mathbf{O}(\sqrt{n}\, m \log(n\mathbf{C}))$ time, where $\mathbf{C} = \max\{|c_{ij}| : (i, j) \in A\}$.

In the weighted version of the inverse assignment problem, the resulting x^0-centered dual inverse problem is a minimum cost flow problem and can be efficiently solved using an efficient minimum cost flow algorithm.

9. THE INVERSE MINIMUM CUT PROBLEM UNDER L_1 NORM

In this section, we study the inverse minimum s-t cut problem. The solution for the inverse minimum cut path problem is readily obtained from the solution to a minimum s-t cut problem, but it is one in which certain arcs of the original network maybe deleted. Although the solution technique is quite straightforward, the proof that the technique is valid is much more difficult than the proofs of the previous two sections. This algorithm was developed by Zhang and Cai (1998). Here we give a much simpler proof than theirs. In particular, we show that the correctness of the algorithm for the minimum cut problem can be obtained by using the path formulation of the cut problem and relying on results from §6.

Consider a connected network $\mathbf{G} = (N, A)$ where u_{ij}s denote arc capacities and s and t are two specified nodes, called the *source* and *sink* nodes, respectively. We assume that $u_{ij} > 0$ for each $(i, j) \in A$. In the network G, we define an *s-t disconnecting set* as a set of arcs whose deletion disconnects the network into two or more components such that nodes s and t belong to different components. We define an *s-t cut* as an s-t disconnecting set whose no proper subset is an s-t disconnecting set. This minimality property implies that in deleting the arcs in an s-t cut creates exactly two components with nodes s and t in different components. Let $\mathbf{S}$ and $\overline{\mathbf{S}}$ (with $\overline{\mathbf{S}} = N - \mathbf{S}$) denote the sets of nodes in the components defined by an s-t

cut; we assume that $s \in \mathbf{S}$ and $t \in \overline{\mathbf{S}}$. We represent this s-t cut as $[\mathbf{S}, \overline{\mathbf{S}}]$. Let $(\mathbf{S}, \overline{\mathbf{S}})$ denote the set of *forward arcs* in the cut, that is, $(\overline{\mathbf{S}}, \mathbf{S}) = \{(i, j) \in A: i \in \mathbf{S} \text{ and } j \in \overline{\mathbf{S}}\}$ and $(\overline{\mathbf{S}}, \mathbf{S})$ denote the set of *backward arcs* in the cut, that is, $(\overline{\mathbf{S}}, \mathbf{S}) = \{(i, j) \in A: i \in \overline{\mathbf{S}} \text{ and } j \in \mathbf{S}\}$. We define the *capacity* of the s-t cut $[S, \overline{\mathbf{S}}]$ as the sum of the capacities of the forward arcs in the cut and denote it by $u[\mathbf{S}, \overline{\mathbf{S}}]$, that is, $u[\mathbf{S}, \overline{\mathbf{S}}] = \sum_{(i, j) \in (\mathbf{S}, \overline{\mathbf{S}})} u_{ij}$. The *minimum cut problem* is to determine an s-t cut of minimum capacity. In the inverse minimum cut problem we are given an s-t cut $[\mathbf{S}^0, \overline{\mathbf{S}}^0]$ that we wish to make a minimum cut by perturbing the arc capacities.

It is well known that the minimum cut problem is equivalent to the dual of the maximum flow problem and can be solved by using standard maximum flow algorithms. Let x^* denote a maximum flow in the network $\mathbf{G}$, and let $\mathbf{S}$ denote the set of nodes reachable from the source node using augmenting paths. Then, $[\mathbf{S}, \overline{\mathbf{S}}]$ is a minimum cut in $\mathbf{G}$ (see, for example, Ahuja et al. 1993).

The minimum cut problem can be formulated as a linear programming problem in several ways. We will use the formulation from which the inverse problem is easier to obtain. We associate a variable y_{ij} for each arc $(i, j) \in A$ whose value is 1 or 0, depending on whether the arc is a forward arc in the minimum cut or not. We denote by $\mathbf{C}(\mathbf{G})$ the collection of all directed paths from node s to node t in the network $\mathbf{G}$. The minimum cut problem can be formulated as the following linear program:

$$\text{Minimize} \sum_{(i, j) \in A} u_{ij} y_{ij}, \tag{26a}$$

subject to

$$\sum_{(i, j) \in \mathbf{P}} y_{ij} \geqslant 1 \quad \text{for all } \mathbf{P} \in \mathbf{C}(\mathbf{G}), \tag{26b}$$

$$0 \leqslant y_{ij} \leqslant 1 \quad \text{for all } (i, j) \in A. \tag{26c}$$

We point out that the upper bound constraints on y_{ij}s are redundant because any optimal solution would automatically satisfy these constraints; however, for simplicity of exposition, we prefer to impose those constraints. If we eliminate the upper bound constraints, then the dual of Equations (26) can easily be shown to be the path flow formulation of the maximum flow problem (see, for example, Ford and Fulkerson 1962). It is well known that there always exists an integer (in fact, a 0-1) optimal solution of Equations (26).

A feasible solution y is minimal for Equations (26) if there is no other feasible solution y' with $y' \leqslant y$. There is a one-to-one correspondence between integer 0-1 solutions of Equations (26) and s-t minimum cuts in $\mathbf{G}$. For any s-t cut $[\mathbf{S}, \overline{\mathbf{S}}]$, setting $y_{ij} = 1$ for each arc $(i, j) \in (\mathbf{S}, \overline{\mathbf{S}})$ and $y_{ij} = 0$ for each $(i, j) \notin (\mathbf{S}, \overline{\mathbf{S}})$ gives a solution y of cost $u[\mathbf{S}, \overline{\mathbf{S}}]$ satisfying Equations (26). If a directed path from node s to node t can contain $p > 1$ forward arcs from the set $(\mathbf{S}, \overline{\mathbf{S}})$, then it must contain $p - 1$ backward arcs from the set $(\overline{\mathbf{S}}, \mathbf{S})$. Further, notice that every feasible 0-1 solution y of Equations (26) gives an s-t disconnecting set, but an optimal solution of (26) must be an s-t cut because arc capacities are strictly positive.

We will now consider the unit-weight inverse minimum cut problem, where we wish to make the cut $[\mathbf{S}^0, \overline{\mathbf{S}}^0]$ a minimum cut by modifying the arc capacities. The formulation (26) is a special case of the 0-1 linear programming problem and its x^0-centered dual inverse problem is the same as (26), except that we eliminate the nonbinding constraints in (26b) with respect to the s-t cut $[\mathbf{S}^0, \overline{\mathbf{S}}^0]$. If the path $\mathbf{P} \in \mathbf{C}(\mathbf{G})$ contains no backward arcs in the cut $[\mathbf{S}^0, \overline{\mathbf{S}}^0]$, then it has exactly one forward arc in the cut $[\mathbf{S}^0, \overline{\mathbf{S}}^0]$, and the constraint in (26b) for path $\mathbf{P}$ is binding. If the path $\mathbf{P} \in \mathbf{C}(\mathbf{G})$ has $p \geqslant 1$ backward arcs in the cut $[\mathbf{S}^0, \overline{\mathbf{S}}^0]$, then it contains $p + 1$ forward arcs in the cut $[\mathbf{S}^0, \overline{\mathbf{S}}^0]$, and the constraint in (26b) for path $\mathbf{P}$ is nonbinding. Let $\mathbf{G}' = (\mathbf{N}, A')$ denote the directed graph obtained by deleting the backward arcs in the cut $[\mathbf{S}^0, \overline{\mathbf{S}}^0]$, that is, $A' = A \backslash [\mathbf{S}^0, \overline{\mathbf{S}}^0]$. Let $\mathbf{C}(\mathbf{G}')$ denote the set of all directed paths from node s to node t in $\mathbf{G}'$. We can thus state the inverse minimum cut problem as

$$\text{Minimize} \sum_{(i, j) \in A} u_{ij} y_{ij}, \tag{27a}$$

subject to

$$\sum_{(i, j) \in \mathbf{P}} y_{ij} \geqslant 1 \quad \text{for all } \mathbf{P} \in \mathbf{C}(\mathbf{G}'), \tag{27b}$$

$$0 \leqslant y_{ij} \leqslant 1 \quad \text{for all } (i, j) \in A', \tag{27c}$$

which is the formulation of the minimum cut problem in the graph $\mathbf{G}'$. We can determine the minimum cut in $\mathbf{G}'$ by solving a maximum flow problem in it. Let x^* denote the maximum flow in $\mathbf{G}'$ and $[\mathbf{S}^*, \overline{\mathbf{S}}^*]$ denote a minimum cut in $\mathbf{G}'$. We can determine the optimal cost vector $\mathbf{d}^*$ for the inverse minimum cut problem using Equation (15), which requires the determination of arc reduced costs. We will now explain how to determine these reduced costs. Let $f_{\mathbf{P}}$ denote the dual variable associated with the constraint in (27b) for the path $\mathbf{P}$; this dual variable corresponds to the flow sent along the path $\mathbf{P}$ in the dual of Equations (27), which is a maximum flow problem in the graph $\mathbf{G}'$. Then the reduced cost of the variable y_{ij}, which we denote by u^f_{ij}, is $u^f_{ij} = u_{ij} - \sum_{\{\mathbf{P} \in \mathbf{C}(i, j)\}} f_{\mathbf{P}}$, where $\mathbf{C}(i, j)$ denote the set of all paths in $\mathbf{C}(\mathbf{G}')$ which contain arc (i, j). But notice that $\sum_{\{\mathbf{P} \in \mathbf{C}(i, j)\}} f_{\mathbf{P}} = x^*_{ij}$, the flow on arc (i, j) in the flow x^*. Hence $u^f_{ij} = u_{ij} - x^*_{ij}$, which is the unused capacity of arc (i, j) in the flow x^*. Substituting this value of reduced costs in Equations (27) yields:

$$d^*_{ij} = \begin{cases} u_{ij} + (u_{ij} - x^*_{ij}) & \text{for each arc } (i, j) \in (\mathbf{S}^*, \overline{\mathbf{S}}^*) \backslash (\mathbf{S}^0, \overline{\mathbf{S}}^0), \\ u_{ij} - (u_{ij} - x^*_{ij}) & \text{for each arc } (i, j) \in (\mathbf{S}^0, \overline{\mathbf{S}}^0) \backslash (\mathbf{S}^*, \overline{\mathbf{S}}^*), \\ u_{ij} & \text{for every other arc} (i, j). \end{cases} \tag{28}$$

We now note that for each arc $(i, j) \in (\mathbf{S}^*, \overline{\mathbf{S}}^*)$, $u_{ij} = x^*_{ij}$ (because each forward arc in the minimum cut must have flow equal to its capacity). Substituting this result in Equation (28) yields for arc (i, j):

$$d^*_{ij} = \begin{cases} x^*_{ij} & \text{for each arc}(i, j) \in (\mathbf{S}^0, \overline{\mathbf{S}}^0) \backslash (\mathbf{S}^*, \overline{\mathbf{S}}^*), \\ u_{ij} & \text{for each arc}(i, j) \notin (\mathbf{S}^0, \overline{\mathbf{S}}^0) \backslash (\mathbf{S}^*, \overline{\mathbf{S}}^*). \end{cases} \quad (29)$$

In other words, to make the cut $[\mathbf{S}^0, \overline{\mathbf{S}}^0]$ a minimum cut, we decrease the capacity of each forward arc (i, j) in the cut $[\mathbf{S}^0, \overline{\mathbf{S}}^0]$ to x^*_{ij}. This ensures that each forward arc in the cut $[\mathbf{S}^0, \overline{\mathbf{S}}^0]$ has flow equal to its capacity. Further, because the cut $[\mathbf{S}^0, \overline{\mathbf{S}}^0]$ has no backward arcs in $\mathbf{G}'$, the cut $[\mathbf{S}^0, \overline{\mathbf{S}}^0]$ is a minimum cut in $\mathbf{G}$. To summarize, we have shown that the inverse minimum cut problem reduces to solving a minimum cut problem that can be solved using any maximum flow algorithm. Currently, the fastest strongly polynomial bound to solve the minimum cut problem (and the maximum flow problem) is $\mathbf{O}(nm \log(n^2/m))$ and is due to Goldberg and Tarjan (1986). The best weakly polynomial bound to solve the maximum flow problem is $\mathbf{O}(\min\{n^{2/3}, m^{1/2}\} m \log(n^2/m) \log \mathbf{U})$, from Goldberg and Rao (1998), where $\mathbf{U} = \max\{u_{ij} : (i, j) \in A\}$.

We now consider the weighted inverse minimum cut problem. Using the same approach as used for the unit weight case, the weighted inverse minimum cut problem can be formulated as the following linear programming problem:

$$\text{Minimize} \sum_{(i, j) \in A} u_{ij} y_{ij}, \quad (30a)$$

subject to

$$\sum_{(i, j) \in \mathbf{P}} y_{ij} \geqslant 1 \quad \text{for all } \mathbf{P} \in \mathbf{C}(\mathbf{G}'), \quad (30b)$$

$$0 \leqslant y_{ij} \leqslant w_{ij} \quad \text{for all}(i, j) \text{ such that } y^0_{ij} = 0, \quad (30c)$$

$$1 - w_{ij} \leqslant y_{ij} \leqslant 1 \quad \text{for all } (i, j) \text{ such that } y^0_{ij} = 1. \quad (30d)$$

It can be shown that the dual of Equations (30) is a minimum cost flow problem. Hence the weighted inverse minimum cut problem can be solved by using a minimum cost flow algorithm.

10. THE INVERSE MINIMUM COST FLOW PROBLEM UNDER L_1 NORM

In this section, we study the inverse version of the minimum cost flow problem. The solution for the minimum cost flow problem is obtained from the solution to a related minimum cost flow problem. This result was first established by Zhang and Liu (1996). Here we show that the result follows directly from the results of §5.

The minimum cost flow problem in a network $\mathbf{G} = (N, A)$ concerns determining the least cost shipment that meets the demands at some nodes of the network by the available supplies at some other nodes. In the minimum cost flow problem, each arc $(i, j) \in A$ has an associated cost c_{ij} and an associated capacity u_{ij}, and each node i has an associated supply/demand $b(i)$. If $b(i) \geqslant 0$, then node i is a supply node; otherwise it is a demand node. We will assume in this section that for any node pair (i, j) both (i, j) and (j, i) do not belong to A. The minimum cost flow problem can be formulated as the following linear programming problem:

$$\text{Minimize} \sum_{(i, j) \in A} c_{ij} x_{ij}, \quad (31a)$$

subject to

$$\sum_{\{j: (i, j) \in A\}} x_{ij} - \sum_{\{j: (j, i) \in A\}} x_{ji} = b(i) \quad \text{for all } i \in N, \quad (31b)$$

$$0 \leqslant x_{ij} \leqslant u_{ij} \quad \text{for all } (i, j) \in A. \quad (31c)$$

In the inverse minimum cost flow problem, we are given a feasible solution x^0 of Equations (31) that we wish to make optimal by perturbing the arc costs. Using the solution x^0, we partition the arc set A into the following three subsets L, U, and F as follows: $L := \{(i, j) \in A: x^0_{ij} = 0\}$, $U := \{(i, j) \in A: x^0_{ij} = u_{ij}\}$, $F := \{(i, j) \in A: 0 < x^0_{ij} < u_{ij}\}$. Then it follows from our discussion in §6 that the 0-centered dual inverse problem of Equations (31) is the following linear programming problem:

$$\text{Minimize} \sum_{(i, j) \in A} c_{ij} y_{ij}, \quad (32a)$$

subject to

$$\sum_{\{j: (i, j) \in A\}} y_{ij} - \sum_{\{J: (j, i) \in A\}} y_{ij} = 0 \quad \text{for all } i \in N, \quad (32b)$$

$$y_{ij} \geqslant 0 \quad \text{for all } (i, j) \in L, \quad (32c)$$

$$y_{ij} \leqslant 0 \quad \text{for all } (i, j) \in U, \quad (32d)$$

$$-1 \leqslant y_{ij} \leqslant 1 \quad \text{for all } (i, j) \in A. \quad (32e)$$

The constraints (32c)–(32e) can alternatively be stated as follows:

$$0 \leqslant y_{ij} \leqslant 1 \quad \text{for all } (i, j) \in L, \quad (33a)$$

$$-1 \leqslant y_{ij} \leqslant 1 \quad \text{for all } (i, j) \in F, \quad (33b)$$

$$-1 \leqslant y_{ij} \leqslant 0 \quad \text{for all } (i, j) \in U. \quad (33c)$$

We can convert the above linear programming problem into a standard minimum cost flow problem (that is, where all variables have a zero lower bound on arc flows) by performing the transformation of variables: (i) for each arc $(i, j) \in L$, replace the variable y_{ij} by the variable y'_{ij} defined as $y'_{ij} = y_{ij}$, with cost $c'_{ij} = c_{ij}$; (ii) for each arc $(i, j) \in U$, replace the variable y_{ij} by the variable y'_{ji} defined as $y'_{ji} = -y_{ij}$, with cost $c'_{ji} = -c_{ij}$; and (iii) for each arc $(i, j) \in F$, replace the variable y_{ij} by the two variable y'_{ij} and y_{ji} defined as $y_{ij} = y'_{ij} - y'_{ji}$, with cost $c'_{ij} = c_{ij}$ and $c'_{ji} = -c_{ij}$. Each variable y'_{ij} is required to be nonnegative.

Let $A(x^0)$ denote the index set of the variables y'_{ij}s. In terms of the variables y'_{ij}s, the inverse minimum cost flow problem can be reformulated as the following linear programming problem:

$$\text{Minimize} \sum_{(i,j)\in A(x^0)} c'_{ij} y'_{ij}, \tag{34a}$$

subject to

$$\sum_{\{j:(i,j)\in A(x^0)\}} y'_{ij} - \sum_{\{j:(j,i)\in A(x^0)\}} y'_{ji} = 0 \quad \text{for all } i \in N, \tag{34b}$$

$$0 \leqslant y'_{ij} \leqslant 1 \quad \text{for all } (i,j) \in A(x^0). \tag{34c}$$

Now observe that Problem (34) is the formulation of the *minimum cost circulation problem* (that is, the minimum cost flow problem with zero supply/demand vector) on a unit capacity network. Further, the network on which the minimum cost circulation problem is solved is known as the residual network of $\mathbf{G}$ corresponding to the flow x^0, where all arc residual capacities are set to one.

The minimum cost flow problem (34) is in general easier to solve than the original minimum cost flow problem (31) because all arc capacities in it are one. Using the successive shortest path algorithm, this minimum cost circulation problem can be solved in $\mathbf{O}(m(m + n\log n))$ time (see, for example, Ahuja et al. 1993). Using the cost scaling algorithm from Gabow and Tarjan (1989), this minimum cost circulation problem can be solved in $\mathbf{O}(\min\{n^{5/3}, m^{3/2}\}\log(n\mathbf{C}))$ time, where $\mathbf{C} = \max\{|c_{ij}| : (i,j) \in A\}$.

We now explain how to obtain the optimal cost vector d^*. Let π denote the optimal dual variables associated with (34b), and $c^{\pi}_{ij} = c_{ij} - \pi_i + \pi_j$ denote the optimal reduced costs. It follows from our discussion in §6 that the optimal cost vector d* is given by Equation (15).

For the weighted version of the inverse minimum cost flow problem, we get the same formulation as Problem (34) except that the constraints (34c) are replaced by the following constraint:

$$0 \leqslant y'_{ij} \leqslant w_{ij} \quad \text{for all } (i,j) \in A(x^0). \tag{34c'}$$

The resulting problem is again a minimum cost flow problem but, in general, all arcs do not have unit capacities. Hence the resulting minimum cost circulation problem cannot be solved as efficiently as in the case of unit capacities.

11. THE INVERSE MINIMUM COST FLOW PROBLEM UNDER L_∞ NORM

We will now apply our results for the minimax inverse linear programming problem to the minimum cost flow problem. Our results also apply to the assignment problem and the shortest path problem as special cases.

In the minimax inverse minimum cost flow problem, we are given a feasible solution x^0 of the minimum cost flow problem (31), which we wish to make optimal by perturbing the arc costs in a manner so that the maximum perturbation is minimum. We assume that $w_j = 1$ for all $j \in J$. In the solution x^0, we partition the arc set A into the following three subsets L, U, and F, as follows: $L := \{(i,j) \in A: x^0_{ij} = 0\}$, $U := \{(i,j) \in A: x^0_{ij} = u_{ij}\}$, $F := \{(i,j) \in A: 0 < x^0_{ij} < u_{ij}\}$. It follows from §6 that the 0-centered minimax dual inverse problem of (31) is the following linear programming problem:

$$\text{Minimize} \sum_{(i,j)\in A} c_{ij} y_{ij}, \tag{35a}$$

subject to

$$\sum_{\{j:(i,j)\in A\}} y_{ij} - \sum_{\{j:(j,i)\in A\}} y_{ji} = 0 \quad \text{for all } i \in N, \tag{35b}$$

$$y_{ij} \geqslant 0 \quad \text{for all } (i,j) \in L, \tag{35c}$$

$$y_{ij} \leqslant 0 \quad \text{for all } (i,j) \in U, \tag{35d}$$

$$\sum_{(i,j)\in A} |y_{ij}| = 1. \tag{35e}$$

We now perform the same transformation of variables as we did in §10, where we replace y_{ij}s by the nonnegative variables y'_{ij}s. The minimum cost flow problem after this transformation can be simplified to the following linear program:

$$\text{Minimize} \sum_{(i,j)\in A(x^0)} c'_{ij} y'_{ij}, \tag{36a}$$

subject to

$$\sum_{\{j:(i,j)\in A(x^0)\}} y'_{ij} - \sum_{\{j:(j,i)\in A(x^0)\}} y'_{ji} = 0 \quad \text{for all } i \in N, \tag{36b}$$

$$\sum_{(i,j)\in A(x^0)} y'_{ij} = 1, \tag{36c}$$

$$y'_{ij} \geqslant 0 \quad \text{for all } (i,j) \in A(x^0). \tag{36d}$$

It is well known that Problem (36) is the formulation of the minimum mean cycle problem (see, for example, Dantzig et al. 1966). A *minimum mean cycle* in the residual network $\mathbf{G}(x^0)$ is a directed cycle W for which the mean cost given by $\sum_{(i,j)\in W} c_{ij}/|W|$ is minimum. We can obtain a minimum mean cycle in $\mathbf{G}(x^0)$ using an algorithm from Karp (1978), which runs in $\mathbf{O}(nm)$ time, or using the algorithm of Orlin and Ahuja (1992), which runs in $\mathbf{O}(\sqrt{nm}\log(n\mathbf{C}))$ time, where $\mathbf{C} = \max\{c_{ij}: (i,j) \in A\}$. Let π denote the vector of optimal dual variables of (7.2) and $c^{\pi}_{ij} = c_{ij} - \pi_i + \pi_j$ denote the optimal reduced costs. A minimum mean cycle algorithm yields the mean cost of the minimum mean cycle and the vector π of optimal dual variables. The optimal cost vector d* can be obtained using Equations (17).

For the weighted case, we get the same formulation as in Problem (36) except that (36c) is replaced by $\sum_{(i,j)\in G(x^0)}(y_{ij}/w_{ij}) \leqslant 1$. This is the formulation of the minimum cost-to-weight ratio cycle problem, which is

also known as the *tramp steamer problem* (see, for example, Dantzig et al. 1966). The minimum cost-to-weight ratio problem is to identify a directed cycle W in the network for which $(\sum_{(i,j)\in W} c_{ij})/(\sum_{(i,j)\in W} w_{ij})$ is minimum. The minimum cost-to-weight ratio problem can be solved in $\mathbf{O}(nm\log(\mathbf{CW}))$ time using Lawler's (1966) algorithm, or in $\mathbf{O}(n^4 \log n)$ time using Meggido's (1979) algorithm, where $\mathbf{C} = \max\{c_{ij}: (i,j) \in A\}$ and $\mathbf{W} = \max\{w_{ij}: (i,j) \in A\}$. It can also be solved in $\mathbf{O}(\sqrt{n}m\log^2(\mathbf{CW}))$ time using Goldberg's (1995) shortest path algorithm.

12. THE GENERAL INVERSE OPTIMIZATION PROBLEM

In this section, we consider the general inverse optimization problem and show (under reasonable regularity conditions) that if the problem **P** is polynomially solvable, then its inverse versions under L_1 and L_∞ norms are also polynomially solvable. This result makes use of the ellipsoid algorithm, and we refer the reader to the books by Schrijver (1986) and Grotschel et al. (1986).

Let **S** denote the set of feasible solutions, and $\mathbf{P} = \min\{cx: x \in \mathbf{S}\}$. We denote by Q^n, the set of all rational numbers in the n dimensional space. Suppose that a polyhedron $\mathbf{D} \subseteq R^n$ is defined by rational linear inequalities in terms of rationals of size at most φ. On the polyhedron **D**, the *separation problem* and *optimization problem*, can be defined as follows.

Separation Problem. Given a vector $\mathrm{d}' \in R^n$, the separation problem is to either decide that $d' \in \mathbf{D}$, or find a vector $y \in Q^n$ such that $dy < d'y$ for all $d \in \mathbf{D}$.

Optimization Problem. Given a polyhedron $\mathbf{D} \subseteq R^n$ and a vector $r \in Q^n$, conclude with one of the following: (a) give a vector $\mathrm{d}^* \in \mathbf{D}$ with $\mathrm{rd}^* = \min\{\mathrm{rd}: d \in \mathbf{D}\}$; (b) assert that there exists a sequence of vectors in d with objective function values unbounded from below; or (c) assert that d is empty;

If **D** is specified by a set of linear constraints, then to solve the separation problem it is sufficient to check whether the given solution d satisfies all the constraints. If yes, then $d' \in \mathbf{D}$; otherwise, a violated constraint gives a "separator vector" y satisfying $dy < d'y$ for all $d \in \mathbf{D}$. We also use the following well known result:

Theorem 1 (Grotschel et al. 1986). *The optimization problem can be solved in time, polynomially bounded by n, φ, the size of r, and the running time of the separation algorithm.*

The inverse linear programming problem **P** under the L_1 norm can be formulated as

$$\text{Minimize } \sum_{j\in J} z_j, \tag{37a}$$

subject to

$$dx^0 \leqslant dx \quad \text{for all } x \in \mathrm{S}, \tag{37b}$$

$$d_j - c_j \leqslant z_j \quad \text{for all } j \in J, \tag{37c}$$

$$c_j - dj \leqslant z_j \quad \text{for all } j \in J. \tag{37d}$$

Observe that the constraints in (37c) and (37d) imply that $z_j \geqslant |d_j - c_j|$. Further, because we minimize $\sum_{j\in J} z_j$, each z_j will equal $|d_j - c_j|$ in an optimum solution. Let $\overline{\mathbf{D}}$ denote the polyhedron defined by the feasible solutions of Problem (37). We assume that all data in the problem are rational and the largest number in the data has size φ. Given a proposed solution (d', z') of (37), we can easily check in linear time whether the solution (d', z') satisfies (37c) and (37d). To check whether the solution d' satisfies (37b), we solve the problem **P** with d' as the cost vector. Let x' denote the resulting optimal solution. If $\mathrm{d}'x^0 \leqslant d'x'$ (in fact, $\mathrm{d}'x^0 = \mathrm{d}'x'$), then d' satisfies (37b); otherwise, we have found a violated inequality $\mathrm{d}'x^0 > \mathrm{d}'x'$. Thus we can solve the separation problem by solving a single instance of **P**. This result in view of Theorem 1 implies that inverse **P** under the L_1 norm is polynomially solvable.

For the minimax inverse **P** we wish to minimize $\max\{|d_j - c_j| : j \in J\}$, subject to $dx^0 \leqslant dx$, for all $x \in \mathrm{S}$. This mathematical program can be formulated as to minimize z subject to Problem (37). Using the same technique as in the case of the L_1 norm, it can be shown that we can solve the separation problem by solving a single instance of problem **P**. We summarize the preceding discussion as the following theorem.

Theorem 2. *If a problem* **P** *is polynomially solvable for each linear cost function, then inverse versions of* **P** *under L_1 and L_∞ are polynomially solvable.*

This result establishes the polynomial solvability of large classes of inverse optimization problems; however, the ellipsoid algorithm is not yet practical for large problems. Moreover, for many specific classes of problems, such as network flow problems, one may obtain improved polynomial time algorithms.

ACKNOWLEDGMENTS

The authors sincerely thank the referees, whose perceptive and insightful comments led to a major improvement of the paper. The authors gratefully acknowledge the support from the Office of Naval Research under contract ONR N00014-98-1-0317, as well as a grant from the United Parcel Service. They also acknowledge the help of Don Wagner, who raised some perceptive and fundamental questions that led to the pursuit of the research reported in this paper.

REFERENCES

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

——, J. B. Orlin. 1998. Combinatorial algorithms for inverse network flow problems. Working Paper, Sloan School of Management, MIT, Cambridge, MA.

——, ——. 2000. A faster algorithm for the inverse spanning tree problem. *J. Algorithms* **34** 177–193.

——, ——. 2001. A fast scaling algorithm for minimizing separable convex functions subject to chain constraints. *Oper. Res.* **49** 784–789.

Burton, D., Ph. L. Toint. 1992. On an instance of the inverse shortest paths problem. *Math. Programming* **53** 45–61.

——, ——. 1994. On the use of an inverse shortest paths algorithm for recovering linearly correlated costs. *Math. Programming* **63** 1–22.

Carr, S. C., W. S. Lovejoy. 1997. The inverse newsvendor problem: Choosing an optimal demand portfolio for capacitated resources. Technical Report, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI.

Dantzig, G. B., W. Blattner, M. R. Rao. 1966. Finding a cycle in a graph with minimum cost to time ratio with application to a ship routing problem. *Theory of Graphs. International Symposium*. Dunod, Paris, and Gordon and Breach, New York, 209–213.

Dembo, R., L. Merkoulovitch, D. Rosen. 1998. Images from a portfolio. Algorithmics Research Working Paper, Algorithmics, Inc., Canada.

Dial, B. 1997. Minimum-revenue congestion pricing, Part 1: A fast algorithm for the single-origin case. Technical Report, The Volpe National Transportation Systems Center, Kendall Square, Cambridge, MA.

Ford, L. R., Jr., D. R. Fulkerson. 1962. *Flows in Networks*. Princeton University Press, Princeton, NJ.

Fredman, M. L., R. E. Tarjan. 1984. Fibonacci heaps and their uses in improved network optimization algorithms. *Proc. 25th IEEE Symposium on Foundations of Comput. Sci.* 338–346.

Gabow, H. N., R. E. Tarjan. 1989. Faster scaling algorithms for network problems. *SIAM J. Comput.* **18** 1013–1036.

Goldberg, A. V. 1995. Scaling algorithms for the shortest path problem. *SIAM J. Comput.* **24** 494–504.

——, S. Rao. 1998. Beyond the flow decomposition barrier. *J. ACM* **45** 753–782.

——, R. E. Tarjan. 1986. A new approach to the maximum flow problem. *Proc. the 18th ACM Symposium on the Theory of Comput.* 136-146. (Full paper in *J. of ACM* **35**(1990), 873–886.)

Goldfarb, D. 1985. Efficient dual simplex algorithms for the assignment problem. *Math. Programming* **33** 187–203.

Greenberg, H. J. 1997. An annotated bibliography for post-solution analysis in mixed integer programming and combinatorial optimization. *Advances in Computational and Stochastic Optimization, Logic Programming, and Heuristic Search*. D. L. Woodruff, ed. Kluwer Academic Publishers, New York.

Grotschel, M., L. Lovasz, A. Schrijver. 1986. *Geometric Algorithms and Combinatorial Optimization*. Springer, Heidelberg.

Karp, R. M. 1978. A characterization of the minimum cycle mean in a diagraph. *Discrete Math.* **23** 309–311.

Meggido, N. 1979. Combinatorial optimization with rational objective functions. *Math. Oper. Res.* **4** 414–424.

Lawler, E. L. 1966. Optimal cycles in doubly weighted linear graphs. *Theory of Graphs: International Symposium*, Dunod, Paris, and Gordon and Breach, New York. 209–213.

Neumann-Denzau, G., J. Behrens. 1984. Inversion of seismic data using tomographical reconstruction techniques for investigations of laterally inhomogeneous media. *Geophysical J. Royal Astronomical Soc.* **79** 305–315.

Nolet, G. 1987. *Seismic Tomography*. Reidel, Dordrecht.

Orlin, J. B., R. K. Ahuja. 1992. New scaling algorithms for the assignment and minimum cycle mean problems. *Math. Programming* **54** 41–56.

Schrijver, A. 1986. *Theory of Linear and Integer Programming*. John Wiley & Sons, New York.

Shan, Y. 1999. Personal communication. CSX Transportation, Jacksonville, FL.

Sheffi, Y. 1985. *Urban Transportation Networks*. MIT, Cambridge, MA.

Sokkalingam, P. T., R. K. Ahuja, J. B. Orlin. 1999. Solving inverse spanning tree problems through network flow techniques. *Oper. Res.* **47** 291–300.

Tarantola, A. 1987. *Inverse Problem Theory: Methods for Data Fitting and Model Parameter Estimation*. Elsevier, Amsterdam.

Woodhouse, J. H., A. M. Dziewonski. 1984. Mapping the upper mantle: Three dimensional modeling of Earth structure by inversion of seismic waveforms. *J. Geophysical Res.* **89** (B7) 5953–5986.

Xu, S., J. Zhang. 1995. An inverse problem of the weighted shortest path problem. *Japanese J. Indust. Appl. Math.* **12** 47–59.

Yang, C., J. Zhang, Z. Ma. 1997. Inverse maximum flow and minimum cut problem. *Optimization* **40** 147–170.

Zhang, J., M. Cai. 1998. Inverse problem of minimum cuts. *ZOR Math. Methods Oper. Res.* **48** 51–58.

——, Z. Liu. 1996. Calculating some inverse linear programming problem. *J. Comput. Appl. Math.* **72** 261–273.