Inverse Optimization

Ravindra K. Ahuja
Industrial and Systems Engineering Department
University of Florida,
Gainesville, FL 32611, USA
ahuja@ufl.edu

James B. Orlin
Sloan School of Management
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
jorlin@mit.edu

(Revised April 30, 2001)

Inverse Optimization

Ravindra K. Ahuja[1] and James B. Orlin[2]

ABSTRACT

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 $\|d - c\|_p = \sum_{j \in J} w_j |d_j - c_j|$, 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 non-unit 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 non-unit 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 non-unit 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] Industrial and Systems Engineering Department, University of Florida, Gainesville, FL 32611, USA.

[2] Sloan School of Management, MIT, Cambridge, MA 02139, USA.

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. These parameters may not all 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 since 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 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, **P** = min{cx : x ∈ **S**}. Suppose that $x^0 \in$ **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, **P**(d) = min{dx: x ∈ **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 $\|d - c\|_p = [\sum_{j \in J} |d_j - c_j|^p]^{1/p}$ 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 Section 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 due to 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 non-unit 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 non-unit 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 non-unit 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, Zhang and Ma [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, Ahuja and Orlin [1999], Ahuja and Orlin [1998a]) and the inverse sorting problem (Ahuja and Orlin [1997]). Ahuja and Orlin [1998b] 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 briefly describe 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, since 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 in a way 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 [1992, 1994] studied nonlinear cost inverse path problems, and Dial [1997] 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 finds applications in portfolio optimization (Carr and Lovejoy [1997] and Dembo, Merkoulovitch and Rosen [1998]). It also has some

plausible applications in isotonic regression (Ahuja and Orlin [1997]), 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**:

$$\text{Minimize } \sum_{j \in J} c_j x_j, \tag{3.1a}$$

subject to

$$\sum_{j \in J} a_{ij} x_j \geq b_i, \qquad \text{for all } i \in I, \tag{3.1b}$$

$$x_j \geq l_j, \qquad \text{for all } j \in J, \tag{3.1c}$$

$$-x_j \geq -u_j, \qquad \text{for all } j \in J, \tag{3.1d}$$

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, u_j denotes the upper bound on x_j. Let us associate the dual variable π_i with the ith constraint in (3.1b), λ_j with the jth constraint in (3.1c), and φ_j with the jth constraint in (3.1d). The dual of **LP** is the following linear program:

$$\text{Maximize } \sum_{i \in I} b_i \pi_i + \sum_{j \in J} l_j \lambda_j - \sum_{j \in J} u_j \varphi_j \tag{3.2a}$$

subject to

$$\sum_{i \in I} a_{ij} \pi_i + \lambda_j - \varphi_j = c_j, \text{ for all } j \in J, \tag{3.2b}$$

$$\pi_i \geq 0, \text{ for all } i \in I; \lambda_j \geq 0 \text{ for all } j \in J; \text{ and } \varphi_j \geq 0 \text{ for all } j \in J. \tag{3.2c}$$

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 (3.1), (π, λ, φ) is feasible for (3.2), and together they satisfy the following complementary slackness conditions:

(i)   if $\sum_{j \in J} a_{ij} x_j > b_i$ then $\pi_i = 0,$ \hfill (3.3a)

(ii) if $x_j > l_j$ then $\lambda_j = 0,$ \hfill (3.3b)

(iii) if $x_j < u_j$ then $\varphi_j = 0.$ \hfill (3.3c)

Let x^0 be a feasible solution of (3.1). Let B denote the index set of binding constraints in (3.1b) with respect to x^0 (that is, $B = \{i \in I: \sum_{j \in J} a_{ij} x_j = b_i\}$), L denote the index set of binding constraints in (3.1c) (that is, $L = \{j \in J: x_j^0 = l_j\}$), and U denote the index set of binding constraints in (3.1d) (that is, $U = \{j \in J: x_j^0 = u_j\}$). Let $F = \{j \in J: 0 < x_j^0 < u_j\}$. We can restate the complementary slackness conditions (3.3) as:

(i) $\pi_i = 0$ for all $i \notin B$,   (3.4a)

(ii) $\lambda_j = 0$ for all $j \notin L$,   (3.4b)

(iii) $\varphi_j = 0$ for all $j \notin U$.   (3.4c)

We want to make x^0 an optimal solution of (3.1) by perturbing the cost vector c to d. We denote by **LP**(d) the linear program (3.1) where the c_j's have been replaced with d_j's. 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 (3.2b) with c_j replaced by d_j and the primal-dual pair satisfies the complementary slackness conditions (3.4). Using (3.4) in (3.2) gives us the following characterization of inverse feasible cost vectors:

$$\sum_{i \in B} a_{ij} \pi_i + \lambda_j = d_j, \text{ for all } j \in L, \qquad\qquad (3.5a)$$

$$\sum_{i \in B} a_{ij} \pi_i - \varphi_j = d_j, \text{ for all } j \in U, \qquad\qquad (3.5b)$$

$$\sum_{i \in B} a_{ij} \pi_i = d_j, \text{ for all } j \in F, \qquad\qquad (3.5c)$$

$\pi_i \geq 0$ for all $i \in B$; $\lambda_j \geq 0$ for all $j \in L$; and $\varphi_j \geq 0$ for all $j \in U$.   (3.5d)

The inverse problem is to minimize $\|d - c\|_p$ over all inverse feasible cost vectors d (that is, all cost vectors d satisfying (3.5)). In the next two sections, we will consider the following two objective functions for the inverse problem: (i) minimize $\sum_{j \in J} w_j |d_j - c_j|$, and (ii) minimize $\max\{w_j|d_j - c_j|: j \in J\}$.

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 imporantly, 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 we establish here 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 $\sum_{j \in J} w_j |d_j - c_j|$, subject to (3.5). 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 $w_j|d_j - c_j|$ is equivalent to minimizing $\alpha_j + \beta_j$, subject to $d_j - c_j = \alpha_j - \beta_j$, $\alpha_j \geq 0$ and $\beta_j \geq 0$. Using this transformation, the inverse linear programming problem can be stated as follows:

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

or, equivalently,

Maximize $-\sum_{j\in J} w_j\alpha_j - \sum_{j\in J} w_j\beta_j,$ (4.1a)

subject to

$$\sum_{i\in B} a_{ij}\,\pi_i - \alpha_j + \beta_j + \lambda_j = c_j, \qquad \text{for all } j \in L, \tag{4.1b}$$

$$\sum_{i\in B} a_{ij}\,\pi_i - \alpha_j + \beta_j - \varphi_j = c_j, \qquad \text{for all } j \in U, \tag{4.1c}$$

$$\sum_{i\in B} a_{ij}\,\pi_i - \alpha_j + \beta_j \quad\;\; = c_j, \qquad \text{for all } j \in F, \tag{4.1d}$$

$\pi_i \geq 0$ for all $i \in B$; $\alpha_j \geq 0$ and $\beta_j \geq 0$ for all $j \in J,$ (4.1e)

$\lambda_j \geq 0$ for all $j \in L$; and $\varphi_j \geq 0$ for all $j \in U.$ (4.1f)

We will now simplify (4.1). We first note that in an optimal solution of (4.1) both α_j and β_j cannot take positive values, since otherwise we can reduce both by a small amount δ without violating any constraint and strictly improving the objective function value. We can restate (4.1b), (4.1c) and (4.1d) as

$$-\alpha_j + \beta_j = c_j^\pi - \lambda_j, \qquad \text{for all } j \in L, \tag{4.2a}$$

$$-\alpha_j + \beta_j = c_j^\pi + \varphi_j, \qquad \text{for all } j \in U, \tag{4.2b}$$

$$-\alpha_j + \beta_j = c_j^\pi, \qquad \text{for all } j \in F, \tag{4.2c}$$

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 non-negativity 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 (4.1), then the optimal cost vector d^* is given by

$$d_j^* = \begin{cases} c_j - |c_j^\pi| & \text{if } c_j^\pi > 0 \text{ and } x_j^0 > 0, \\ c_j + |c_j^\pi| & \text{if } c_j^\pi < 0 \text{ and } x_j^0 < u_{ij}, \\ c_j & \text{otherwise.} \end{cases} \qquad (4.3)$$

We have shown above that we can solve the inverse problem by solving (4.1), and the optimal values of π can be used to obtain the optimal cost vector using (4.3). Instead of solving (4.1), we can alternatively solve the dual of (4.1) which turns out to be a variation of the original problem (3.1). We associate the variable y_j with the constraint associated with the jth index in (4.1b) to (4.1d) and then take its dual. We get the following linear programming problem:

$$\text{Minimize } \sum\nolimits_{j \in J} c_j y_j, \qquad (4.4a)$$

subject to

$$\sum\nolimits_{j \in J} a_{ij}\, y_j \geq 0, \qquad \text{for all } i \in B, \qquad (4.4b)$$
$$0 \leq y_j \leq w_j, \qquad \text{for all } j \in L, \qquad (4.4c)$$
$$-w_j \leq y_j \leq 0, \qquad \text{for all } j \in U, \qquad (4.4d)$$
$$-w_j \leq y_j \leq w_j, \qquad \text{for all } j \in F. \qquad (4.4e)$$

We can formulate the dual inverse problem in an alternate manner that may be more convenient to work with compared to the formulation in (4.4). Substituting $y_j = x_j - x_j^0$ for each $j \in J$ in (4.4) gives us the following equivalent formulation that is more similar to the original formulation of **LP**:

$$\text{Minimize } \sum\nolimits_{j \in J} c_j x_j, \qquad (4.5a)$$

subject to

$$\sum\nolimits_{j \in J} a_{ij}\, x_j \geq b_i, \qquad \text{for all } i \in B, \qquad (4.5b)$$
$$0 \leq x_j \leq w_j, \qquad \text{for all } j \in L, \qquad (4.5c)$$
$$u_j - w_j \leq x_j \leq u_j, \qquad \text{for all } j \in U, \qquad (4.5d)$$
$$x_j^0 - w_j \leq x_j \leq x_j^0 + w_j, \qquad \text{for all } j \in F. \qquad (4.5e)$$

The formulations (4.4) and (4.5) 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 (4.4) as *the 0-centered dual inverse problem*, and to the formulation (4.5) 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 "$\geq$". In case we have some "$\leq$" inequalities in (3.1), then in the 0-centered problem or the x^0-centered problem the corresponding constraint (if binding) will also be a "$\leq$" inequality. In case we have an equality

constraint in (3.1), 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 where 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 which 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. Since x^0 is a 0-1 solution, each index $j \in J$ either belongs to L or U, and in both the cases (4.5c) and (4.5d) reduce to $0 \le x_j \le 1$. We thus get the following x^0-centered dual inverse problem:

$$\text{Minimize } \sum_{j \in J} c_j x_j \tag{5.1a}$$

subject to

$$\sum_{j \in J} a_{ij} x_j \ge b_i, \qquad \text{for all } i \in B, \tag{5.1b}$$

$$0 \le x_j \le 1, \qquad \text{for all } j \in J, \tag{5.1c}$$

which is the same as the original problem except that the non-binding constraints with respect to x^0 have been eliminated. In the case when all constraints are binding (for example, when each constraint in (5.1b) 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 (5.1) and π denote the optimal dual variables associated with the constraints in (5.1b). 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 (4.5) yields the following optimal cost vector:

$$d_j^* = \begin{cases} c_j - |c_j^\pi| & \text{for all } j \text{ satisfying } x_j^0 = 1 \text{ and } x_j^* = 0, \\ c_j + |c_j^\pi| & \text{for all } j \text{ satisfying } x_j^0 = 0 \text{ and } x_j^* = 1, \\ c_j & \text{for all } j \text{ satisfying } x_j^0 = x_j^*. \end{cases} \tag{5.2}$$

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

9

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 $w_j \geq 0$ for all $j \in J$. It follows from the discussion in Section 3 that the minimax inverse linear programming problem is to minimize $\max\{w_j|d_j - c_j|: j \in J\}$ subject to (3.5). This mathematical program is not a linear program since 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 $d_j - c_j = \alpha_j - \beta_j$, $\alpha_j \geq 0$ and $\beta_j \geq 0$. Further, to eliminate the maximization of the terms, we introduce a nonnegative variable θ and add the constraints $w_j\alpha_j + w_j\beta_j \leq \theta$ 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:

$$\text{Maximize } -\theta \tag{6.1a}$$

subject to

$$\sum_{i\in B} a_{ij} \pi_i - \alpha_j + \beta_j + \lambda_j = c_j, \quad \text{for all } j \in L, \tag{6.1b}$$

$$\sum_{i\in B} a_{ij} \pi_i - \alpha_j + \beta_j - \varphi_j = c_j, \quad \text{for all } j \in U, \tag{6.1c}$$

$$\sum_{i\in B} a_{ij} \pi_i - \alpha_j + \beta_j \quad\quad = c_j, \quad \text{for all } j \in F, \tag{6.1d}$$

$$w_j\alpha_j + w_j\beta_j - \theta \leq 0, \quad\quad\quad \text{for all } j \in J, \tag{6.1e}$$

$$\pi_i \geq 0 \text{ for all } i \in B; \alpha_j \geq 0 \text{ and } \beta_j \geq 0 \text{ for all } j \in J, \tag{6.1f}$$

$$\lambda_j \geq 0 \text{ for all } j \in L; \text{ and } \varphi_j \geq 0 \text{ for all } j \in U. \tag{6.1g}$$

Let $c_j^\pi = c_j - \sum_{i\in B} a_{ij} \pi_i$. We can restate (6.1b), (6.1c) and (6.1d) as

$$-\alpha_j + \beta_j = c_j^\pi - \lambda_j, \quad\quad \text{for all } j \in L, \tag{6.2a}$$

$$-\alpha_j + \beta_j = c_j^\pi + \varphi_j, \quad\quad \text{for all } j \in U, \tag{6.2b}$$

$$-\alpha_j + \beta_j = c_j^\pi, \quad\quad\quad\quad \text{for all } j \in F, \tag{6.2c}$$

There are three cases to consider.

Case 1. $c_j^\pi > 0$. The non-negativity 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 (6.1b), (6.1c), (6.1d) and (6.1e) reduce to $w_j c_j^\pi \le \theta$ 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 (6.1b), (6.1c), (6.1d) and (6.1e) reduce to $-w_j c_j^\pi \le \theta$ 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 (6.1b), (6.1c), (6.1d) and (6.1e) remain satisfied.

The previous case analysis also implies that the optimal cost vector $d_j^* = c_j + \alpha_j - \beta_j$ is given by

$$
d_j^* = \begin{cases}
c_j - |c_j^\pi| & \text{if } c_j^\pi > 0 \text{ and } x_j^0 > 0, \\
c_j + |c_j^\pi| & \text{if } c_j^\pi < 0 \text{ and } x_j^0 < u_{ij}, \\
c_j & \text{otherwise.}
\end{cases}
\tag{6.3}
$$

which is the same as in the case of L_1 norm. The preceding analysis also allows us to formulate (6.2) as the following linear program:

$$\text{Maximize } -\theta, \tag{6.4a}$$

subject to

$$-w_j(c_j - \sum_{i \in B} a_{ij} \pi_i) \le \theta, \qquad \text{for all } j \in F \cup L. \tag{6.4b}$$

$$w_j(c_j - \sum_{i \in B} a_{ij} \pi_i) \le \theta, \qquad \text{for all } j \in F \cup U, \tag{6.4c}$$

$$\pi_i \ge 0 \text{ for all } i \in B, \tag{6.4d}$$

which can be reformulated as

$$\text{Maximize } -\theta, \tag{6.5a}$$

subject to

$$\sum_{i \in B} a_{ij} \pi_i - \frac{1}{w_j} \theta \le c_j \qquad \text{for all } j \in F \cup L. \tag{6.5b}$$

$$-\sum_{i \in B} a_{ij} \pi_i - \frac{1}{w_j} \theta \le -c_j \qquad \text{for all } j \in F \cup U. \tag{6.5c}$$

$$\pi_i \ge 0 \text{ for all } i \in B, \tag{6.5d}$$

where for simplicity of exposition we assume that $w_j \neq 0$ for each $j \in J$. By taking the dual of (6.5), we can obtain an equivalent formulation of the minimax inverse problem. We associate the variable y_j^+ with the constraint (6.5b), and the variable y_j^- with the constraint (6.5c). The dual of (6.5) is the following linear programming problem:

$$\text{Minimize } \sum\nolimits_{j \in L} c_j y_j^+ + \sum\nolimits_{j \in F} c_j (y_j^+ - y_j^-) - \sum\nolimits_{j \in U} c_j y_j^- \tag{6.6}$$

subject to

$$\sum\nolimits_{j \in L} a_{ij} y_j^+ + \sum\nolimits_{j \in F} a_{ij} (y_j^+ - y_j^-) - \sum\nolimits_{j \in U} a_{ij} y_j^- \geq 0 \text{ for all } i \in B, \tag{6.6b}$$

$$\sum\nolimits_{j \in L} \frac{1}{w_j} y_j^+ + \sum\nolimits_{j \in F} \frac{1}{w_j} (y_j^+ + y_j^-) + \sum\nolimits_{j \in U} \frac{1}{w_j} y_j^- = 1, \tag{6.6c}$$

$$y_j^+ \geq 0, \quad y_j^- \geq 0 \text{ for all } j \in J. \tag{6.6d}$$

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_j's, we can reformulate (6.6) as follows:

$$\text{Minimize } \sum\nolimits_{j \in J} c_j y_j, \tag{6.7a}$$

subject to

$$\sum\nolimits_{j \in J} a_{ij} y_j \geq 0, \text{ for all } i \in B, \tag{6.7b}$$

$$\sum\nolimits_{j \in J} \frac{1}{w_j} |y_j| = 1, \tag{6.7c}$$

$$y_j \geq 0 \text{ for all } j \in L, \text{ and } y_j \leq 0 \text{ for all } j \in U. \tag{6.7d}$$

We have assumed in (6.7) that $w_j \neq 0$ for each $j \in J$. It is easy to see that for the case when some w_j have zero values, the formulation will be identical to that in (6.7) 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 (6.7b), then the optimal cost vector d^* can be computed using (6.4). We refer to the formulation (6.7) 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 (6.7). This gives us the following equivalent formulation of the minimax inverse problem:

$$\text{Minimize } \sum\nolimits_{j \in J} c_j x_j \tag{6.8a}$$

subject to

$$\sum\nolimits_{j \in J} a_{ij} x_j \geq b_i, \qquad \text{for all } i \in B, \tag{6.8b}$$

$$\sum_{j\in J} \frac{1}{w_j} |x_j - x_j^0| \le 1. \tag{6.8c}$$

$$x_j \ge x_j^0 \text{ for all } j \in L, \text{ and } x_j \le x_j^0 \text{ for all } j \in U. \tag{6.8d}$$

We have assumed so far in this section that that all inequalities are of the form "≥". In case we have some "≤" inequalities, then in the 0-centered problem or the x^0-centered problem the corresponding constraint (if binding) will also be a "≤" 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 mininimax 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 Section 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 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:

$$\text{Minimize } \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{7.1a}$$

subject to

$$\sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = \begin{cases} 1 & \text{for } i = s, \\ 0 & \text{for all } i \notin \{s,t\}, \\ -1 & \text{for } i = t, \end{cases} \tag{7.1b}$$

$$0 \le x_{ij} \le 1 \quad \text{for all } (i, j) \in A. \tag{7.1c}$$

We assume that the network G does not contain any negative cost cycle; under this assumption (7.1) is a 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 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 P^0, that is, $x_{ij}^0 = 1$ for all $(i, j) \in P^0$ and $x_{ij}^0 = 0$ for all $(i, j) \notin P^0$. The shortest path problem is a special case of the 0-1 linear programming problem, and it follows from our discussion in Section 5 that the unit-weight x^0-centered dual inverse shortest path problem is identical to (7) because all constraints in (7.1b) are binding constraints. Let 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, Magnanti and Orlin [1993]) that reduced costs corresponding to the shortest path P^* satisfy the following conditions:

$$c_{ij}^\pi = 0, \text{ for all } (i, j) \in P^*, \tag{7.2a}$$

$$c_{ij}^\pi \geq 0, \text{ for all } (i, j) \notin P^*. \tag{7.2b}$$

Let $P^*\backslash P^0 = \{(i, j) \in A: (i, j) \in P^* \text{ and } (i, j) \notin P^0\}$, and $P^0\backslash P^* = \{(i, j) \in A: (i, j) \in P^0 \text{ and } (i, j) \notin P^*\}$. Using these results in (5.2) gives the following optimal cost vector d^*:

$$d_{ij}^* = \begin{cases} c_{ij} - c_{ij}^\pi & \text{for all } (i, j) \in P^0 \backslash P^* \\ c_{ij} & \text{for all } (i, j) \notin P^0 \backslash P^* \end{cases}, \tag{7.3}$$

In words, the above result implies that for each arc which is in P^0 but not in 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 P^0 by $\Sigma_{(i,j) \in P0\backslash P^*} c_{ij}^\pi$ units and does not affect the cost of the path P^*. After this change, the modified reduced cost of each arc (i, j) in 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 non-negative, 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, Magnanti and Orlin [1993]), or in $O(\sqrt{n} \, m \log \mathbf{C})$ 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 efficiently 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 Section 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 Section 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 Section 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:

$$\text{Minimize } \sum_{(i,j) \in A} c_{ij} x_{ij} \tag{8.1a}$$

subject to

$$\sum_{\{j:(i,j) \in A\}} x_{ij} = 1 \qquad \text{for all } i \in N_1, \tag{8.1b}$$

$$-\sum_{\{i:(i,j) \in A\}} x_{ij} = -1 \quad \text{for all } i \in N_2, \tag{8.1c}$$

$$0 \le x_{ij} \le 1 \qquad \text{for all } (i, j) \in A. \tag{8.1d}$$

Each 0-1 solution x of (8.1) defines an assignment $M = \{(i, j) \in A: x_{ij} = 1\}$. Conversely, each assignment M defines a solution x of (8.1). 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 (5.1) and its x^0-centered dual inverse problem for the unit-weight case is the same as (8.1). 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 $c_{ij}^{\pi} \ge 0$ for all $(i, j) \notin M^*$. Using this result in (5.2) gives us the following optimal cost vector d^* for the inverse assignment problem:

$$d_{ij}^* = \begin{cases} c_{ij} - c_{ij}^{\pi} & \text{for all } (i, j) \in M^0 \setminus M^* \\ c_{ij} & \text{for all } (i, j) \notin M^0 \setminus M^* \end{cases} \tag{8.2}$$

Currently, the best available strongly polynomial time bound to solve the assignment problem is $O(nm + n^2 \log n)$ 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 $O(\sqrt{n} \, m \, \log(n\mathbf{C}))$ 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 may be 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 Section 6.

Consider a connected network $G = (N, A)$ 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 S and $\overline{S}$ (with $\overline{S} = N - S$) denote the sets of nodes in the components defined by an s-t cut; we assume that $s \in S$ and $t \in \overline{S}$. We represent this s-t cut as $[S,\overline{S}]$. Let $(S,\overline{S})$ denote the set of *forward arcs* in the cut, that is, $(S,\overline{S}) = \{(i, j) \in A : i \in S \text{ and } j \in \overline{S} \}$ and $(\overline{S},S)$ denote the set of *backward arcs* in the cut, that is, $(\overline{S},S) = \{(i, j) \in A : i \in \overline{S} \text{ and } j \in S\}$. We define the *capacity* of the s-t cut $[S,\overline{S}]$ as the sum of the capacities of the forward arcs in the cut and denote it by $u[S,\overline{S}]$, that is, $u[S,\overline{S}] = \sum_{(i,j)\in(S,\overline{S})} u_{ij}$. 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 $[S^0,\overline{S}^0]$ which 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 G, and let S denote the set of nodes reachable from the source node using augmenting paths. Then, $[S,\overline{S}]$ is a minimum cut in G (see, for example, Ahuja, Magnanti and Orlin [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 upon whether the arc is a forward arc in the minimum cut or not. We denote by $\mathbf{C}(G)$ the collection of all directed paths from node s to node t in the network G. The minimum cut problem can be formulated as the following linear program:

$$\text{Minimize} \sum_{(i,j)\in A} u_{ij}\, y_{ij} \tag{9.1a}$$

subject to

$$\sum_{(i,j)\in P} y_{ij} \geq 1, \quad \text{for all } P \in \mathbf{C}(G), \tag{9.1b}$$

$$0 \leq y_{ij} \leq 1, \qquad \text{for all } (i, j) \in A. \tag{9.1c}$$

We point out that the upper bound constraints on y_{ij}'s are redundant since 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 (9.1) can be easily 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 (9.1).

A feasible solution y is minimal for (9.1) if there is no other feasible solution y' with $y' \leq y$. There is a one-to-one correspondence between integer 0-1 solutions of (9.1) and s-t minimum cuts in G. For any s-t cut $[S, \overline{S}]$, setting $y_{ij} = 1$ for each arc $(i, j) \in (S, \overline{S})$ and $y_{ij} = 0$ for each $(i, j) \notin (S, \overline{S})$ gives a solution y of cost $u[S, \overline{S}]$ satisfying (9.1). If a directed path from node s to node t can contains $p > 1$ forward arcs from the set $(S, \overline{S})$, then it must contain p-1 backward arcs from the set $(\overline{S}, S)$. Further, notice that every feasible 0-1 solution y of (9.1) gives an s-t disconnecting set, but an optimal solution of (9.1) 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 $[S^0, \overline{S}^0]$ a minimum cut by modifying the arc capacities. The formulation (9.1) is a special case of the 0-1 linear programming problem and its x^0-centered dual inverse problem is the same as (9.1) except that we eliminate the non-binding constraints in (9.1b) with respect to the s-t cut $[S^0, \overline{S}^0]$. If the path $P \in$ $\mathbf{C}(G)$ contains no backward arcs in the cut $[S^0, \overline{S}^0]$, then it has exactly one forward arc in the cut $[S^0, \overline{S}^0]$, and the constraint in (9.1b) for path P is binding. If the path $P \in \mathbf{C}(G)$ has $p \geq 1$ backward arcs in the cut $[S^0, \overline{S}^0]$, then it contains p+1 forward arcs in the cut $[S^0, \overline{S}^0]$, and the constraint in (9.1b) for path P is non-binding. Let $G' = (N, A')$ denote the directed graph obtained by deleting the backward arcs in the cut $[S^0, \overline{S}^0]$, that is, $A' = A \setminus [S^0, \overline{S}^0]$. Let $\mathbf{C}(G')$ denote the set of all directed paths from node s to node t in G'. We can thus state the inverse minimum cut problem as:

$$\text{Minimize } \sum_{(i,j) \in A} u_{ij} y_{ij} \tag{9.2a}$$

subject to

$$\sum_{(i,j) \in P} y_{ij} \geq 1, \qquad \text{for all } P \in \mathbf{C}(G'), \tag{9.2b}$$

$$0 \leq y_{ij} \leq 1, \qquad \text{for all } (i, j) \in A', \tag{9.2c}$$

which is the formulation of the minimum cut problem in the graph G'. We can determine the minimum cut in G' by solving a maximum flow problem in it. Let x^* denote the maximum flow in G' and $[S^*, \overline{S}^*]$ denote a minimum cut in G'. We can determine the optimal cost vector d^* for the inverse minimum cut problem using (6.3), which requires the determination of arc reduced costs. We will now explain how to determine these reduced costs. Let f_P denote the dual variable associated with the constraint in (9.2b) for the path P; this dual variable corresponds to the flow sent along the path P in the dual of (9.2) which is a maximum flow problem in the graph G'. Then the reduced cost of the variable y_{ij}, which we denote by

u_{ij}^f, is $u_{ij}^f = u_{ij} - \sum_{\{P \in \mathbf{C}(i,j)\}} f_P$, where $\mathbf{C}(i, j)$ denote the set of all paths in $\mathbf{C}(G')$ which contain arc (i, j).

But notice that $\sum_{\{P \in \mathbf{C}(i,j)\}} f_P = x_{ij}^*$, the flow on arc (i, j) in the flow x^*. Hence $u_{ij}^f = u_{ij} - x_{ij}^*$, which is

the unused capacity of arc (i, j) in the flow x^*. Substituting this value of reduced costs in (9.2) yields:

$$d_{ij}^* = \begin{cases} u_{ij} + (u_{ij} - x_{ij}^*) & \text{for each arc } (i, j) \in (S^*, \overline{S}^*) \setminus (S^0, \overline{S}^0) \\ u_{ij} - (u_{ij} - x_{ij}^*) & \text{for each arc } (i, j) \in (S^0, \overline{S}^0) \setminus (S^*, \overline{S}^*) \\ u_{ij} & \text{for every other arc } (i, j). \end{cases} \tag{9.3}$$

We now note that for each arc $(i, j) \in (S^*, \overline{S}^*)$, $u_{ij} = x_{ij}^*$ (because each forward arc in the minimum cut must have flow equal to its capacity). Substituting this result in (9.3) yields for arc (i, j):

$$d_{ij}^* = \begin{cases} x_{ij}^* & \text{for each arc } (i, j) \in (S^0, \overline{S}^0) \setminus (S^*, \overline{S}^*) \\ u_{ij} & \text{for each arc } (i, j) \notin (S^0, \overline{S}^0) \setminus (S^*, \overline{S}^*) \end{cases} . \tag{9.4}$$

In other words, in order to make the cut $[S^0, \overline{S}^0]$ a minimum cut, we decrease the capacity of each forward arc (i, j) in the cut $[S^0, \overline{S}^0]$ to x_{ij}^*. This ensures that each forward arc in the cut $[S^0, \overline{S}^0]$ has flow equal to its capacity. Further, since the cut $[S^0, \overline{S}^0]$ has no backward arcs in G', the cut $[S^0, \overline{S}^0]$ is a minimum cut in 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 $O(nm \log(n^2/m))$ and is due to Goldberg and Tarjan [1986]. The best weakly polynomial bound to solve the maximum flow problem is $O(\min\{n^{2/3}, m^{1/2}\}m \log(n^2/m) \log \mathbf{U})$ and is due to Goldberg and Rao [1997], 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:

$$\text{Minimize } \sum_{(i,j) \in A} u_{ij} y_{ij} \tag{9.5a}$$

subject to

$$\sum_{(i,j) \in P} y_{ij} \geq 1, \text{ for all } P \in \mathbf{C}(G'), \tag{9.5b}$$

$$0 \leq y_{ij} \leq w_{ij}, \text{ for all } (i, j) \text{ such that } y_{ij}^0 = 0, \tag{9.5c}$$

$$1 - w_{ij} \leq y_{ij} \leq 1, \text{ for all } (i, j) \text{ such that } y_{ij}^0 = 1. \tag{9.5d}$$

It can be shown that the dual of (9.5) 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 Section 5.

The minimum cost flow problem in a network $G = (N, A)$ 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 $b(i) \geq 0$, 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:

$$\text{Minimize} \sum_{(i,j)\in A} c_{ij}\, x_{ij} \tag{10.1a}$$

subject to

$$\sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = b(i) \quad \text{for all } i \in N, \tag{10.1b}$$

$$0 \leq x_{ij} \leq u_{ij}, \qquad \text{for all } (i, j) \in A. \tag{10.1c}$$

In the inverse minimum cost flow problem, we are given a feasible solution x^0 of (10.1) which 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: $L := \{(i, j) \in A: x_{ij}^0 = 0\}$, $U := \{(i, j) \in A: x_{ij}^0 = u_{ij}\}$, $F := \{(i, j) \in A: 0 < x_{ij}^0 < u_{ij}\}$. Then it follows from our discussion in Section 6 that the 0-centered dual inverse problem of (10.1) is the following linear programming problem:

$$\text{Minimize} \sum_{(i,j)\in A} c_{ij}\, y_{ij} \tag{10.2a}$$

subject to

$$\sum_{\{j:(i,j)\in A\}} y_{ij} - \sum_{\{j:(j,i)\in A\}} y_{ji} = 0 \qquad \text{for all } i \in N, \tag{10.2b}$$

$$y_{ij} \geq 0, \qquad \text{for all } (i, j) \in L, \tag{10.2c}$$

$$y_{ij} \leq 0, \qquad \text{for all } (i, j) \in U, \tag{10.2d}$$

$$-1 \leq y_{ij} \leq 1, \qquad \text{for all } (i, j) \in A. \tag{10.2e}$$

The constraints (10.2c), (10.2d), and (10.2e) can alternatively be stated as follows:

$$0 \leq y_{ij} \leq 1, \qquad \qquad \text{for all } (i, j) \in L, \tag{10.3a}$$

$$-1 \leq y_{ij} \leq 1, \qquad \qquad \text{for all } (i, j) \in F, \tag{10.3b}$$

$$-1 \leq y_{ij} \leq 0, \qquad \qquad \text{for all } (i, j) \in U. \tag{10.3c}$$

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 non-negative. 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:

$$\text{Minimize} \sum_{(i,j) \in A(x^0)} c'_{ij} \, y'_{ij} \tag{10.4a}$$

subject to

$$\sum_{\{j : (i,j) \in A(x^0)\}} y'_{ij} - \sum_{\{j : (j,i) \in A(x^0)\}} y'_{ji} = 0 \quad \text{for all } i \in N, \tag{10.4b}$$

$$0 \leq y'_{ij} \leq 1, \qquad \qquad \text{for all } (i, j) \in A(x^0). \tag{10.4c}$$

Now observe that (10.4) 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 G corresponding to the flow x^0 where all arc residual capacities are set to one.

The minimum cost flow problem (10.4) is in general easier to solve than the original minimum cost flow problem (10.1) because all arc capacities in it are one. Using the successive shortest path algorithm, this minimum cost circulation problem can be solved in $O(m(m + n \log n))$ time (see, for example, Ahuja, Magnanti, and Orlin [1993]). Using the cost scaling algorithm due to Gabow and Tarjan [1989], this minimum cost circulation problem can be solved in $O(O(\min\{n^{5/3}, m^{3/2}\}\log(n\mathbf{C}))$ 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 (10.4b), and $c^{\pi}_{ij} = c_{ij} - \pi_i + \pi_j$ denote the optimal reduced costs. It follows from our discussion in Section 6 that the optimal cost vector d^* is given by (6.3).

For the weighted version of the inverse minimum cost flow problem, we get the same formulation as (10.4) except that the constraints (10.4) are replaced by the following constraint:

$$0 \le y'_{ij} \le w_{ij}, \qquad \text{for all } (i, j) \in A(x^0). \tag{10.4c'}$$

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 (10.1) 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 Section 6 that the 0-centered minimax dual inverse problem of (10.1) is the following linear programming problem:

$$\text{minimize } \sum_{(i,j) \in A} c_{ij} \, y_{ij} \tag{11.1a}$$

subject to

$$\sum_{\{j:(i,j) \in A\}} y_{ij} - \sum_{\{j:(j,i) \in A\}} y_{ji} = 0 \qquad \text{for all } i \in N, \tag{11.1b}$$

$$y_{ij} \ge 0, \qquad\qquad \text{for all } (i, j) \in L, \tag{11.1c}$$

$$y_{ij} \le 0, \qquad\qquad \text{for all } (i, j) \in U, \tag{11.1d}$$

$$\sum_{(i,j) \in A} |y_{ij}| = 1. \tag{11.1e}$$

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

$$\text{Minimize } \sum_{(i,j) \in A(x^0)} c'_{ij} \, y'_{ij} \tag{11.2a}$$

subject to

$$\sum_{\{j:(i,j) \in A(x^0)\}} y'_{ij} - \sum_{\{j:(j,i) \in A(x^0)\}} y'_{ji} = 0 \quad \text{for all } i \in N, \tag{11.2b}$$

$$\sum_{(i,j) \in A(x^0)} y'_{ij} = 1, \tag{11.2c}$$

$$y'_{ij} \ge 0, \qquad\qquad \text{for all } (i, j) \in A(x^0). \tag{11.2d}$$

21

It is well known that (11.2) is the formulation of the minimum mean cycle problem (see, for example, Dantzig, Blattner and Rao [1966]). A *minimum mean cycle* in the residual network $G(x^0)$ 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 $G(x^0)$ using an algorithm due to Karp [1978] which runs in O(nm) time, or using the algorithm due to Orlin and Ahuja [1992] which runs in $O(\sqrt{n}\, m\, \log(n\mathbf{C}))$ time, where $\mathbf{C} = \max\{c_{ij}$: $(i, j) \in A\}$. Let π denote the vector of optimal dual variables of (7.2) and $c_{ij}^{\pi} = c_{ij} - \pi_i + \pi_j$ 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 (6.5).

For the weighted case, we get the same formulation as in (11.2) except that (11.2c) is replaced by $\sum_{(i,j)\in G(x^0)} (y_{ij}/w_{ij}) \le 1$. 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, Blattner and Rao [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 $O(nm \log(\mathbf{CW}))$ time using Lawler's [1966] algorithm or in $O(n^4 \log n)$ 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 $O(\sqrt{n}\, m \log^2(\mathbf{CW}))$ 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, Lovasz, and Schrijver [1986].

Let **S** the 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 $d' \in R^n$, 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 $d^* \in \mathbf{D}$ with $rd^* = \min\{rd: d \in \mathbf{D}\}$; (b) assert that there exists a sequence of vectors in $\mathbf{D}$ with objective function values unbounded from below; or (c) assert that $\mathbf{D}$ 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, Lovasz, and Schrijver [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:

$$\text{Minimize } \sum_{j \in J} z_j \tag{12.1a}$$

subject to

$$dx^0 \leq dx \qquad \text{for all } x \in \mathbf{S}, \tag{12.1b}$$
$$d_j - c_j \leq z_j \qquad \text{for all } j \in J, \tag{12.1c}$$
$$c_j - d_j \leq z_j \qquad \text{for all } j \in J. \tag{12.1d}$$

Observe that the constraints in (12.1c) and (12.1d) imply that $z_j \geq |d_j - c_j|$. Further, since 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 (12.1). We assume that all data in (12.1) is rational and the largest number in the data has size φ. Given a proposed solution (d', z') of (12.1), we can easily check in linear time whether the solution (d', z') satisfies (12.1c) and (12.1d). To check whether the solution d' satisfies (12.1b), we solve the problem **P** with d' as the cost vector. Let x' denote the resulting optimal solution. If $d'x^0 \leq d'x'$ (in fact, $d'x^0 = d'x'$), then d' satisfies (12.1b); otherwise, we have found a violated inequality $d'x^0 > d'x'$. 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 $dx^0 \leq dx$, for all $x \in \mathbf{S}$. This mathematical program can be formulated as to minimize z subject to (12.1). 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.

REFERENCES

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

Ahuja, R. K., and J. B. Orlin. 1997. A fast scaling algorithm for minimizing separable convex functions subject to chain constraints. Working Paper, Sloan School of Management, MIT, Cambridge, MA.

Ahuja, R. K., and J. B. Orlin. 1998a. A faster algorithm for the inverse spanning tree problem. Working Paper, Sloan School of Management, MIT, Cambridge, MA. To appear in *Journal of Algorithms*.

Ahuja, R. K., and J. B. Orlin. 1998b. Combinatorial algorithms for inverse network flow problems. Working Paper, Sloan School of Management, MIT, Cambridge, MA.

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

Burton, D., and Ph. L. Toint. 1994. On the use of an inverse shortest paths algorithm for recovering linearly correlated costs. *Mathematical Programming* **63**, 1-22.

Carr, S. C., and 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, and M. R. Rao. 1966. Finding a cycle in a graph with minimum cost to time ratio with application to a ship routing problem. In *Theory of Graphs. International Symposium*. Dunod, Paris, and Gordon and Breach, New York, pp. 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 02142.

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

Fredman, M. L., and R. E. Tarjan. 1984. Fibonacci heaps and their uses in improved network optimization algorithms. *Proceedings of the 25th Annual IEEE Symposium on Foundations of Computer Science*, pp. 338-346.

Gabow, H. N., and R. E. Tarjan. 1989. Faster scaling algorithms for network problems. *SIAM Journal on Computing* **18**, 1013-1036.

Goldberg, A. V. 1995. Scaling algorithms for the shortest path problem. *SIAM Journal on Computing* **24**, 494-504.

Goldberg, A. V., and S. Rao. 1997. Length function for flow computation. Technical Report # 97-055, NEC Research Institute, 4 Independence Way, Princeton, NJ.

Goldberg, A. V., and R. E. Tarjan. 1986. A new approach to the maximum flow problem. *Proceedings of the 18th ACM Symposium on the theory of Computing*, pp. 136-146. Full paper in *Journal of ACM* **35**(1990), 873-886.

Goldfarb, D. 1985. Efficient dual simplex algorithms for the assignment problem. *Mathematical 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*, Edited by D. L. Woodruff, Kluwer Academic Publishers.

Grotschel, M., L. Lovasz, and 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 Mathematics* **23**, 309-311./*-*-*-*-***

Meggido, N. 1979. Combinatorial optimization with rational objective functions. *Mathematics of Operations Research* **4**, 414-424.

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

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

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

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

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

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

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

Sokkalingam, P. T., R. K. Ahuja, and J. B. Orlin. 1999. Solving inverse spanning tree problems through network flow techniques. *Operations Research* **47**, 291-300.

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

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

Xu, S., and J. Zhang. 1995. An inverse problem of the weighted shortest path problem. *Japanese Journal of Industrial and Applied Mathematics* **12**, 47-59.

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

Zhang, J., and M. Cai. 1998. Inverse problem of minimum cuts. *ZOR Mathematical Methods of Operations Research* **48**, 51-58.

Zhang, J., and Z. Liu. 1996. Calculating some inverse linear programming problem. *Journal of Computational and Applied Mathematics* **72**, 261-273.