Inverse Optimization, Part II: Network Flow Problems

by
Ravindra K. Ahuja
James B. Orlin

SWP# 4003 February 1998

Inverse Optimization, Part II: Network Flow Problems

Ravindra K. Ahuja*
Sloan School of Management
Massachusetts Institute of Technology
Cambridge, MA 02139, USA

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

(Revised January 25, 1998)

* On leave from Indian Institute of Technology, Kanpur 208 016, INDIA.

Inverse Optimization, Part II: Network Flow Problems

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**, and $||d - c||_p$ is minimum, where $||d - c||_p$ is some selected L_p norm. In Part 1 of this paper, we considered inverse linear programming problems under the L_1 and L_∞ norms. In this paper, we consider the specialization of these results to the following network flow problems: the shortest path problem, the assignment problem, the minimum cut problem, and the minimum cost flow problem. We show that in the case of the L_1 norm, the inverse versions of each of these network flow problems reduce to a problem of the same kind; that is, the inverse shortest path problem reduces to the shortest path problem, the inverse assignment problem reduces to the assignment problem, and so on. We next consider the case of the L_∞ norm, and show that the inverse versions of the shortest path problem, the assignment problem, and the minimum cost flow problem reduce to the minimum mean cycle problem. (The minimum mean cycle problem is a directed cycle whose cost divided by the number of arcs in it is minimum.)

[1] Sloan School of Management, MIT, Cambridge, MA 02139, USA; On leave from Indian Institute of Technology, Kanpur 208 016, INDIA.

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

1. INTRODUCTION

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 let 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 becomes an optimal solution of **P**, and $\|d - c\|_p$ is minimum, where $\|.\|_p$ denotes some selected L_p norm. In Part I of this paper, Ahuja and Orlin [1998a], we considered inverse linear programming problems under a weighted L_1 norm (that is, we minimize $\sum_{j \in J} w_j|d_j - c_j|\}$) and also under a weighted L_∞ norm (that is, we minimize $\max\{w_j|d_j - c_j| : j \in J\}$, where J is the index set of decision variables x. In this paper, we consider the specialization of those results to the following network flow problems: the shortest path problem, the assignment problem, the minimum cut problem, and the minimum cost flow problem. We consider the unit weight as well as general weight cases. In an another paper, Ahuja and Orlin [1998b], we consider combinatorial algorithms for the network flow problems with unit weights and develop combinatorial proofs of correctness.

Inverse optimization is a relatively new area of research. Ahuja and Orlin [1998a] provide various references in the area of inverse optimization and compile several applications. There has already been some research devoted to inverse network flow problems. Burton and Toint [1992, 1994], and Burton, Pulleyblank and Toint [1997] have considered inverse shortest path problems (multi-source, multi-sink problems) under the L_2 norm and solved them using non-linear programming techniques. Cai and Yang [1994], and Xu and Zhang [1995] have considered the inverse shortest path problem under the weighted L_1 norm; Yang and Zhang [1996] have considered the maximum capacity path problems; Huang and Liu [1995] have studied the minimum cost flow problem under the weighted L_1 norm; Yang, Zhang and Ma [1997], and Zhang and Cai [1998] have considered the minimum cut problem under the weighted L_1 norm. Each of these problems reduces to solving a minimum cost flow problem. Sokkalingam [1996] in his doctoral dissertation, under the supervision of the first author, also considered the inverse minimum cost flow problem under the weighted L_1 and weighted L_∞ norms. Our research subsumes most of the previous research on the network flow problems under the L_1 and L_∞ norms. We present a unified approach, provide simpler proofs, and obtain faster algorithms for the unit weight cases.

In this paper, we refer to the inverse version of an optimization problem **P** under the L_1 norm as *inverse* ***P***, and the inverse version of problem **P** under the L_∞ norm as *minimax inverse* ***P***. The major contributions made in this paper are as follows:

1. We show that inverse versions of each of the following network flow problems under the L_1 norm and unit weights reduce to a network flow problem of the same kind: the shortest path problem, the assignment problem, the minimum cut problem, and the minimum cost flow problem; that is, the inverse shortest path problem reduces to the shortest path problem, the inverse assignment problem reduces to the assignment problem, and so on. The weighted case for each of these problems reduces to a minimum cost flow problem.

2. We show that the inverse versions of each of the following network flow problems under the L_∞ norm reduce to the minimum mean cycle problem: the shortest path problem, the assignment problem, and the minimum cost flow problem. (A minimum mean cycle is a directed cycle whose cost divided by the number of arcs in it is minimum among all directed cycles in the network.)

2. PRELIMINARIES

In this paper, we review some results from the inverse linear programming problem proved in Part 1 of this paper, Ahuja and Orlin [1998a]. We summarize these results in this section for the sake of completeness.

Inverse Linear Programming Problem

Consider the following bounded variable linear programming problem **LP**:

$$\text{Minimize } \sum_{j \in J} c_j x_j, \tag{2.1a}$$

subject to

$$\sum_{j \in J} a_{ij} x_j \left\{ \begin{matrix} \geq \\ = \\ \leq \end{matrix} \right\} b_i, \quad \text{for all } i \in I, \tag{2.1b}$$

$$0 \leq x_j \leq u_j, \quad \text{for all } j \in J, \tag{2.1c}$$

where J denotes the index set of decision variables x, and I denotes the index set of constraints. Let x^0 be a feasible solution of **LP**. We call a cost vector d to be *inverse feasible* for **LP** (with respect to the solution x^0) if x^0 is an optimal solution of (2.1) when the cost vector c is replaced by the cost vector d. Inverse **LP** problem under the L_1 norm is to find an inverse feasible cost vector d^* of **LP** for which $\|d^* - c\|_1 = \sum_{j\in J}| d_j^* - c_j|$ is minimum among all inverse feasible vectors d. Inverse **LP** problem under the L_∞ norm is to find an inverse feasible cost vector d^* of **LP** for which $\|d^* - c\|_\infty = \max\{| d_j^* - c_j|: j \in J\}$ is minimum. We call such a vector d^* an *optimal cost vector*. In this paper, we refer to the inverse **LP** problem under the L_1 norm as the *inverse **LP** problem*, and the inverse **LP** problem under the L_∞ norm as *the minimax inverse **LP** problem*. In the first part of the paper, we proved the following result:

Theorem 1. *Let x^0 be a feasible solution of **LP** and $B \subseteq I$ denote the index set of constraints that are binding with respect to x^0. Then the inverse linear programming problem under the L_1 norm is the dual of of the following problems:*

***0-centered dual inverse problem**: Same as **LP** except that we (i) replace the variables x_j by the variables y_j and make the right-hand side of (2.1b) and (2.1c) zero; (ii) we eliminate those constraints in (2.1b) and (2.1c) which are non-binding with respect to the solution x^0; and (iii) add the following constraints: $-1 \le y_j \le 1$ for all $j \in J$.*

***x^0-centered dual inverse problem:** Same as **LP** except that we (i) eliminate those constraints in (2.1b) and (2.1c) which are non-binding with respect to the solution x^0; and (ii) add the following constraints: $x_j^0 - 1 \le x_j \le x_j^0 + 1$ for all $j \in J$.*

Let π denote the optimal dual variables associated with the binding constraints. Then the optimal cost vector d^* is given by

$$d_j^* = \begin{cases} c_j - |c_j^\pi| & \text{if } c_j^\pi > 0 \text{ and } x_j^0 > 0, \\ c_j + |c_j^\pi| & \text{if } c_j^\pi < 0 \text{ and } x_j^0 < u_{ij}, \\ c_j & \text{otherwise.} \end{cases} \qquad (2.2)$$

A generalization of the inverse **LP** problem is the case with the objective function $\Sigma_{j\in J} w_j |d_j - c_j|$, where w_j's are specified constants. We refer to this problem as the *weighted inverse **LP** problem.* The 0-centered dual inverse problem for the weighted version is the same as in the unweighted case except that the constraint $-1 \le y_j \le 1$ is replaced by the constraint $-w_j \le y_j \le w_j$. The x^0-centered problem for the weighted version is the same as the unweighted case except that the constraint $x_j^0 - 1 \le x_j \le x_j^0 + 1$ is replaced by the constraint $x_j^0 - w_j \le x_j \le x_j^0 + w_j$.

Inverse 0-1 Linear Programming Problem

A special case of (2.1) is a linear programming problem where $u_j = 1$ for each $j \in J$, and for which there always exists an optimal solution which is a 0-1 solution. Let x^0 denote a feasible solution of the 0-1 linear programming problem. We have shown in Ahuja and Orlin [1998a] that the x^0-centered dual inverse problem for the 0-1 linear programming program is the following linear programming problem:

$$\text{Minimize } \Sigma_{j\in J} c_j x_j, \tag{2.3a}$$

subject to

$$\Sigma_{j\in J} a_{ij} x_j \left\{\begin{matrix}\ge\\=\\\le\end{matrix}\right\} b_i, \quad \text{for all } i \in B, \tag{2.3b}$$

$$0 \le x_j \le 1, \quad \text{for all } j \in J, \tag{2.3c}$$

where B is the index set of binding constraints. In this case, we can also restate the formula for the optimal cost vector d^*. Let x^* be an optimal solution of the x^0-centered dual inverse problem. If follows from the bounded variable linear programming theory that (i) if $c_j^\pi < 0$ then $x_j^* = u_j = 1$, and (ii) if $c_j^\pi > 0$ then $x_j^* = 0$. Using these results in (2.2) yields the following optimal cost vector:

$$d_j^* = \begin{cases} c_j - |c_j^\pi| & \text{for all j satisfying } x_j^0 = 1 \text{ and } x_j^* = 0, \\ c_j + |c_j^\pi| & \text{for all j satisfying } x_j^0 = 0 \text{ and } x_j^* = 1, \\ c_j & \text{otherwise.} \end{cases} \tag{2.4}$$

Minimax Inverse Linear Programming Problem

In the first part of this paper, we proved the following result:

Theorem 2. *Let x^0 be a feasible solution of **LP** and $B \subseteq I$ denote the index set of constraints that are binding with respect to x^0. Then the inverse linear programming problem under the L_∞ norm is the dual of of the following problems:*

***0-centered minimax dual inverse problem**: Same as **LP** except that we (i) replace the variables x_j by the variables y_j and make the right-hand side of (2.1b) and (2.1c) zero; (ii) eliminate those constraints in (2.1b) and (2.1c) which are non-binding with respect to the solution x^0; and (iii) add the following constraints: $\Sigma_{j \in J} |y_j| \leq 1$.*

***x^0-centered minimax dual inverse problem:** Same as **LP** except that (i) we eliminate those constraints in (2.1b) and (2.1c) which are non-binding with respect to the solution x^0; and (ii) add the following constraints: $\Sigma_{j \in J} |x_j - x_j^0| \leq 1$.*

Let π denote the optimal dual variables associated with the binding constraints. Then the optimal cost vector d^* is given by (2.2), which is the same as that for the L_1 norm.

In the weighted minimax inverse linear programming problem, the objective function is to minimize $\max\{w_j|d_j - c_j| : j \in J\}$, where $w_j \geq 0$ for each $j \in J$. The formulation of the weighted 0-centered minimax dual inverse problem is exactly the same as the formulation for the unit weight case except that the constraint $\Sigma_{j \in J} |y_j| \leq 1$ is replaced by the following constraint: $\Sigma_{\{j \in J \text{ and } w_j \neq 0\}} |y_j|/w_j \leq 1$ and $y_j = 0$ for all j for which $w_j = 0$. The optimal cost vector d^* is computed using (2.2).

3. THE INVERSE SHORTEST PATH PROBLEM

In this section, we study the single-source, single-sink shortest path problem. Inverse shortest path problems have earlier been studied by Burton and Toint [1992, 1994], and Burton, Toint, and Pulleyblank [1997]. They studied multi-source, multi-sink shortest path problems with the L_2 and suggested nonlinear programming techniques to solve those problems. Cai and Yang [1994], Xu and Zhang [1995], and Zhang, Ma and

Yang [1995] have studied various kind of inverse shortest path problems and showed that they reduce to minimum cost flow problems. Dial [1997] has studied the inverse shortest path problem in acyclic networks. In this section, we obtain results which for the weighted L_1 norm are comparable to the results of Cai and Yang [1994] and Xu and Zhang [1995]. In addition, we show that the unit weight inverse shortest path problem is the dual of the shortest path problem and can be solved far more efficiently.

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 (single-source, single-sink) 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:

$$\text{Minimize } \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{3.1a}$$

subject to

$$\sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = \begin{cases} 1 & \text{for } i = s, \\ 0 & \text{for all } i \notin \{s,t\}, \\ -1 & \text{for } i = t, \end{cases} \tag{3.1b}$$

$$0 \le x_{ij} \le 1 \quad \text{for all } (i, j) \in A. \tag{3.1c}$$

We assume that the network G does not contain any negative cost cycle; under this assumption (3.1) has a finite optimal solution. In the inverse shortest path problem, we are given a 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, $x^0_{ij} = 1$ for all (i, j) $\in P^0$ and $x^0_{ij} = 0$ 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 2 that the x^0-centered dual inverse shortest path problem is identical to (3) because all constraints in (3.1b) are binding constraints. Let P^* denote the shortest s-t path in G with c_{ij} as the 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 $c^{\pi}_{ij} = c_{ij} - \pi_i + \pi_j$. It is well known (see, for example, Ahuja, Magnanti and Orlin [1993]) that the reduced costs corresponding to the shortest path P^* satisfy the following conditions:

$$c_{ij}^{\pi} = 0, \qquad \text{for all } (i, j) \in P^*, \text{ and} \tag{3.2a}$$

$$c_{ij}^{\pi} \geq 0, \qquad \text{for all } (i, j) \notin P^*. \tag{3.2b}$$

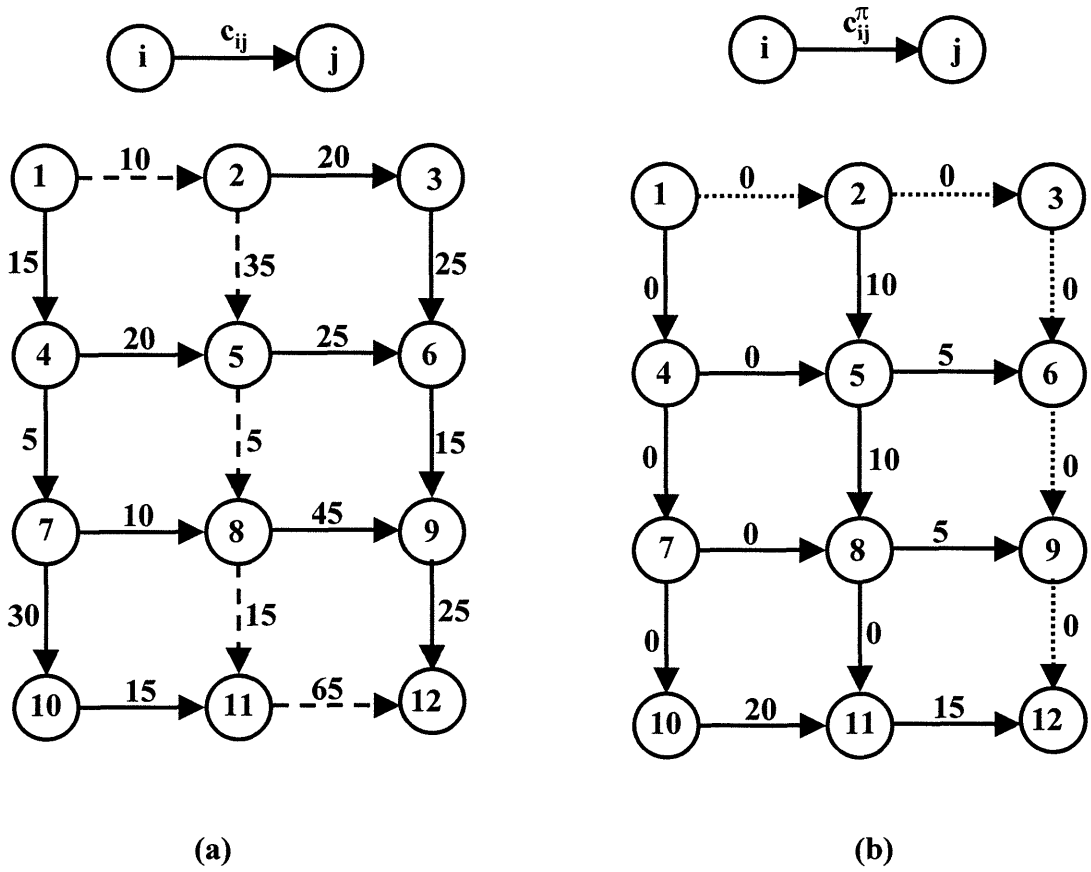
We will now use (2.4) to determine the optimal cost vector d^*. Let $P^* \backslash P^0 = \{(i, j) \in A: (i, j) \in P^* \text{ and } (i, j) \notin P^0\}$, and $P^0 \backslash P^* = \{(i, j) \in A: (i, j) \in P^0 \text{ and } (i, j) \notin P^*\}$. It follows from (3.2a) that $c_{ij}^{\pi} = 0$ for all $(i, j) \in P^*$. Using these results in (2.4) gives the following optimal cost vector d^*:

$$d_{ij}^* = \begin{cases} c_{ij} - c_{ij}^{\pi} & \text{for all } (i, j) \in P^0 \setminus P^* \\ c_{ij} & \text{for all } (i, j) \notin P^0 \setminus P^* \end{cases} \tag{3.3}$$

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 $\sum_{(i,j) \in P^0 \backslash P^*} c_{ij}^{\pi}$ 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 illustrate our algorithm for the inverse shortest path problem using a numerical example. Figure 1 shows a shortest path network where node 1 is the source node, node 12 is the sink node, and P^0 = 1-2-5-8-11-12. We solve a shortest path problem on the network shown in Figure 1(a). The optimal node potentials are given by π = {0, -10, -30, -15, -35, -55, -20, -30, -70, -50, -45, -95}. We point out that optimal node potentials are the negative of the shortest path distances. Figure 1(b) shows the optimal reduced costs. The shortest path in the network is P^* = 1-2-3-6-9-12. Thus $P^0 \backslash P^*$ = {(2, 5), (5, 8), (8, 11), (11, 12)}, and the optimal cost vector d^* is obtained by decreasing the costs of the arcs (2, 5), (5, 8), (8, 11), and (11, 12) by 10, 10, 0, and 15 units, respectively.

We have shown above that the 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



(a) (b)

Figure 1. Illustrating the algorithm for the inverse shortest path problem.

(see, for example, Ahuja, Magnanti and Orlin [1993]), or in $O(\sqrt{n}\, m \log \mathbf{C})$ time using Goldberg's [1995] algorithm, where $\mathbf{C} = \max\{|c_{ij}| : (i, j) \in A\}$.

Weighted Case

In the weighted version of the inverse shortest path problem, the constraint $0 \le x_{ij} \le 1$ in (3.1) is replaced by the constraint $x^0_{ij} - w_{ij} \le x_{ij} \le x^0_{ij} + w_{ij}$. The resulting linear program is no more a shortest path problem. In fact, it is a minimum cost flow problem with possibly negative lower bounds on arc flows. Using standard transformations, negative lower bounds can be replaced by zero lower bounds yielding a standard minimum cost flow problem. Currently, the fastest strongly polynomial time algorithm for the minimum cost flow problem is due to Orlin [1988] and runs in $O(m(m + n \log n) \log n)$ time. The fastest weakly polynomial algorithm for the minimum cost flow problem is $O(\min\{nm \log(n^2/m) \log(n\mathbf{C}), nm(\log\log \mathbf{W}) \log(n\mathbf{C})\}$, where $\mathbf{C} = \max\{|c_{ij}| : (i, j) \in A\}$, and $\mathbf{W} = \max\{w_{ij} : (i, j) \in A\}$; the two time bounds in this expression are due to Goldberg and Tarjan [1987] and Ahuja et al. [1992].

4. THE INVERSE ASSIGNMENT PROBLEM

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:

$$\text{Minimize } \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{4.1a}$$

subject to

$$\sum_{\{j:(i,j)\in A\}} x_{ij} = 1 \quad \text{for all } i \in N_1 \tag{4.1b}$$

$$-\sum_{\{i:(i,j)\in A\}} x_{ij} = -1 \quad \text{for all } j \in N_2 \tag{4.1c}$$

$$0 \le x_{ij} \le 1 \quad \text{for all } (i, j) \in A. \tag{4.1d}$$

Each 0-1 solution x of (4.1) defines an assignment $M = \{(i, j) \in A: x_{ij} = 1\}$. Conversely, each assignment M defines a solution x of (4.1). 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 (2.3) and its x^0-centered dual inverse problem is the same as (4.1). 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 $c_{ij}^{\pi} \geq 0$ for all $(i, j) \notin M^*$. Using this result in (2.4) gives us the following optimal cost vector d^* for the inverse assignment problem:

$$d_{ij}^* = \begin{cases} c_{ij} - c_{ij}^{\pi} & \text{for all } (i,j) \in M^0 \setminus M^* \\ c_{ij} & \text{for all } (i,j) \notin M^0 \setminus M^* \end{cases} \tag{4.2}$$

In words, to make the given assignment M^0 optimal for the assignment problem, we determine an optimal assignment M^* and decrease the cost of each arc $(i, j) \in M^0 \backslash M^*$ by an amount equal to c_{ij}^{π} (the optimal reduced cost of arc (i, j)). This makes the cost of the assignment M^0 equal to that of M^* and M^0 becomes an alternate optimal assignment. 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 it runs in $O(\sqrt{n}\, m \log(n\mathbf{C}))$ time, where $\mathbf{C} = \max\{|c_{ij}| : (i, j) \in A\}$.

We illustrate our algorithm for the inverse assignment problem using the numerical example shown in Figure 2. Let $M^0 = \{(1, 6) (2, 7), (3, 8), (4, 9), (5, 10)\}$ be the assignment that we wish to make optimal by perturbing the arc costs. We solve an assignment problem in the network shown in Figure 2(a). The optimal assignment is M^* = {(1, 7), (2, 6), (3, 8), (4, 10), (5, 9)}, the optimal dual variables are given by π = {0, -5, 15, -10, 25, -10, -10, -20, -5, -25}, and the optimal reduced costs are shown in Figure 2(b). The set $M^0 \backslash M^*$ = {(1, 6), (2, 7), (4, 9), (5, 10)} and the optimal cost vector d^* is obtained by decreasing the costs of arcs {(1, 6), (2, 7), (4, 9), (5, 10) by 0, 15, 10, and 5 units, respectively.

Weighted Case

In the weighted version of the inverse assignment problem, the constraint $0 \leq x_{ij} \leq 1$ in (4.1) is replaced by the constraint $x_{ij}^0 - w_{ij} \leq x_{ij} \leq x_{ij}^0 + w_{ij}$. The resulting linear

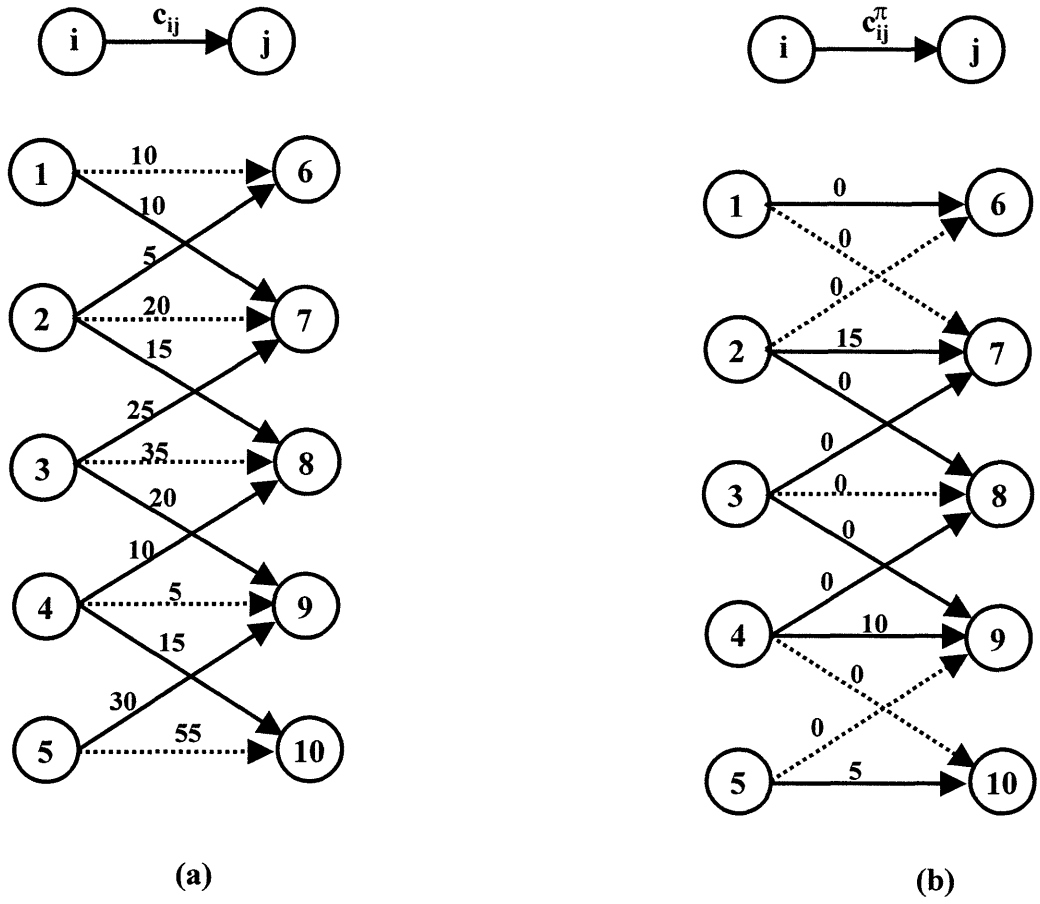


Figure 2. Illustrating the algorithm for the inverse assignment problem.

program is no more an assignment problem. It is a minimum cost flow problem and can be solved using any minimum cost flow algorithm.

5. THE INVERSE MINIMUM CUT PROBLEM

In this section, we study the inverse minimum cut problem. The inverse minimum cut problem has earlier been studied by Yang, Zhang, and Ma [1997], and Zhang and Cai [1998], and they show that the weighted version of the inverse minimum cut problem can be reduced to a minimum cost flow problem. We obtain this result too, and in addition we show that the unit weight version of the inverse problem reduces to solving a minimum cut problem.

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 disconnected 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 $\bar{S}$ (with $\bar{S} = N - S$) denote the sets of nodes in the components defined by an s-t cut; we assume that $s \in S$ and $t \in \bar{S}$. We represent this s-t cut as $[S, \bar{S}]$. Let $(S, \bar{S})$ denote the set of *forward arcs* in the cut, that is, $(S, \bar{S}) = \{(i, j) \in A : i \in S \text{ and } j \in \bar{S}\}$ and $(\bar{S}, S)$ denote the set of *backward arcs* in the cut, that is, $(\bar{S}, S) = \{(i, j) \in A : i \in \bar{S} \text{ and } j \in S\}$. We define the *capacity* of the s-t cut $[S, \bar{S}]$ as the sum of the capacities of the forward arcs in the cut and denote it by $u[S, \bar{S}]$, that is, $u[S, \bar{S}] = \sum_{(i,j) \in (S, \bar{S})} u_{ij}$. 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 $[S^0, \bar{S}^0]$ 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, $[S, \bar{S}]$ 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 linear program:

$$\text{Minimize } \sum_{(i,j)\in A} u_{ij}\, y_{ij} \tag{5.1a}$$

subject to

$$\sum_{(i,j)\in P} y_{ij} \geq 1, \qquad \text{for all } P \in \mathbf{C}(G), \tag{5.1b}$$

$$0 \leq y_{ij} \leq 1, \qquad \text{for all } (i, j) \in A. \tag{5.1c}$$

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 (5.1) can be easily shown to be the path flow formulation of the maximum flow problem (see, for example, Ford and Fulkerson [1962]). The maximum flow minimum cut theorem implies that there always exists an integer (in fact, a 0-1) optimal solution of (5.1).

There is one-to-one correspondence between integer 0-1 solutions of (5.1) and s-t cuts in G. To see this, consider any s-t cut $[S, \bar{S}]$. Setting $y_{ij} = 1$ for each arc (i, j) $\in$ $(S, \bar{S})$ and $y_{ij} = 0$ for each (i, j) $\notin$ $(S, \bar{S})$ gives a solution y of cost $u[S, \bar{S}]$ satisfying (5.1) because every directed path from node s to node t must contain at least one forward arc in $(S, \bar{S})$. Notice that a directed path from node s to node t can contain $p > 1$ forward arcs from the set $(S, \bar{S})$, but then it must contain p-1 backward arcs from the set $(\bar{S}, S)$. Further, notice that every feasible 0-1 solution y of (5.1) gives an s-t disconnecting set, but an optimal solution of (5.1) must be an s-t cut because arc capacities are strictly positive.

We will now consider the inverse minimum cut problem, where we wish to make the cut $[S^0, \bar{S}^0]$ a minimum cut by modifying the arc capacities. The formulation (5.1) is a special case of the 0-1 linear programming problem and its x^0-centered dual inverse problem is the same as (5.1) except that we eliminate the non-binding constraints in (5.1b) with respect to the s-t cut $[S^0, \bar{S}^0]$. If the path P $\in$ $\mathbf{C}(G)$ contains no backward

arcs in the cut $[S^0, \bar{S}^0]$, then it has exactly one forward arc in the cut $[S^0, \bar{S}^0]$, and the constraint in (5.1b) for path P is binding. If the path $P \in \mathbf{C}(G)$ has $p \geq 1$ backward arcs in the cut $[S^0, \bar{S}^0]$, then it contains p+1 forward arcs in the cut $[S^0, \bar{S}^0]$, and the constraint in (5.1b) for path P is non-binding. Let $G' = (N, A')$ denote the directed graph obtained by deleting the backward arcs in the cut $[S^0, \bar{S}^0]$, that is, $A' = A \setminus (\bar{S}^0, S^0)$. 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:

$$\text{Minimize } \sum_{(i,j)\in A} u_{ij}\, y_{ij} \tag{5.2a}$$

subject to

$$\sum_{(i,j)\in P} y_{ij} \geq 1, \qquad \text{for all } P \in \mathbf{C}(G'), \tag{5.2b}$$

$$0 \leq y_{ij} \leq 1, \qquad \text{for all } (i, j) \in A', \tag{5.2c}$$

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 $[S^*, \bar{S}^*]$ denote a minimum cut in G'. We can determine the optimal cost vector d* for the inverse minimum cut problem using (2.4), 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 (5.2b) for the path P; this dual variable corresponds to the flow sent along the path P in the dual of (5.2) which is a maximum flow problem in the graph G'. Then the reduced cost of the variable y_{ij}, which we denote by u^f_{ij}, is $u^f_{ij} = u_{ij} - \sum_{\{P\in\mathbf{C}(i,j)\}} f_P$, 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 $u^f_{ij} = u_{ij} - x^*_{ij}$, which is the unused capacity arc (i, j) in the flow x*. Substituting this value of reduced costs in (5.2) yields:

$$d^*_{ij} = \begin{cases} u_{ij} + (u_{ij} - x^*_{ij}) & \text{for each arc } (i, j) \in (S^*, \bar{S}^*) \setminus (S^0, \bar{S}^0) \\ u_{ij} - (u_{ij} - x^*_{ij}) & \text{for each arc } (i, j) \in (S^0, \bar{S}^0) \setminus (S^*, \bar{S}^*) \\ u_{ij} & \text{for every other arc } (i, j). \end{cases} \tag{5.3}$$

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 (5.3) yields:

$$d^*_{ij} = \begin{cases} x^*_{ij} & \text{for each arc } (i,j) \in (S^0, \overline{S}^0) \setminus (S^*, \overline{S}^*) \\ u_{ij} & \text{for each arc } (i,j) \notin (S^0, \overline{S}^0) \setminus (S^*, \overline{S}^*) \end{cases}. \tag{5.4}$$

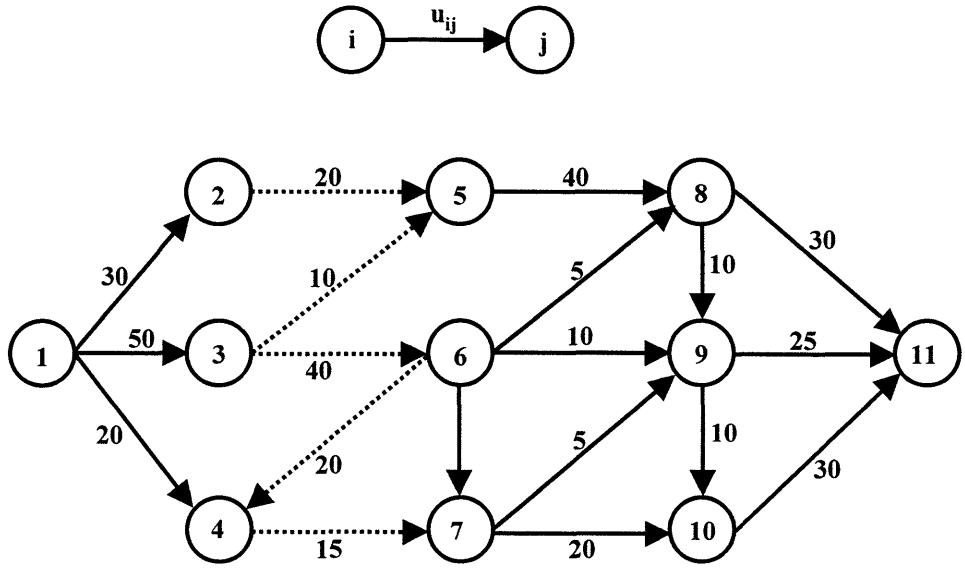
In other words, in order to make the cut $[S^0, \overline{S}^0]$ a minimum cut, we decrease the capacity of each forward arc (i, j) in the cut $[S^0, \overline{S}^0]$ to x^*_{ij}. This ensures that each forward arc in the cut $[S^0, \overline{S}^0]$ has flow equal to its capacity. Further, since the 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.

We illustrate our algorithm for the inverse minimum cut problem using the numerical example shown in Figure 3(a). In the network shown in Figure 3(a), we wish to make the s-t cut $[S^0, \overline{S}^0]$ a minimum cut where $S^0 = \{1, 2, 3, 4\}$. This cut has only one backward arc (6, 4). We delete it from the network and solve a maximum flow problem. Figure 3(b) shows the maximum flow and the minimum cut $[S^*, \overline{S}^*]$ with $S^* = \{1, 2, 3, 4, 6, 7\}$. The optimal capacity vector is obtained by setting the capacities of arcs in $(S^0, \overline{S}^0)$ equal to their flow values; this results in decreasing the capacity of the arc (3, 6) from 40 to 30 and the capacity of the arc (4, 7) from 15 to 10.

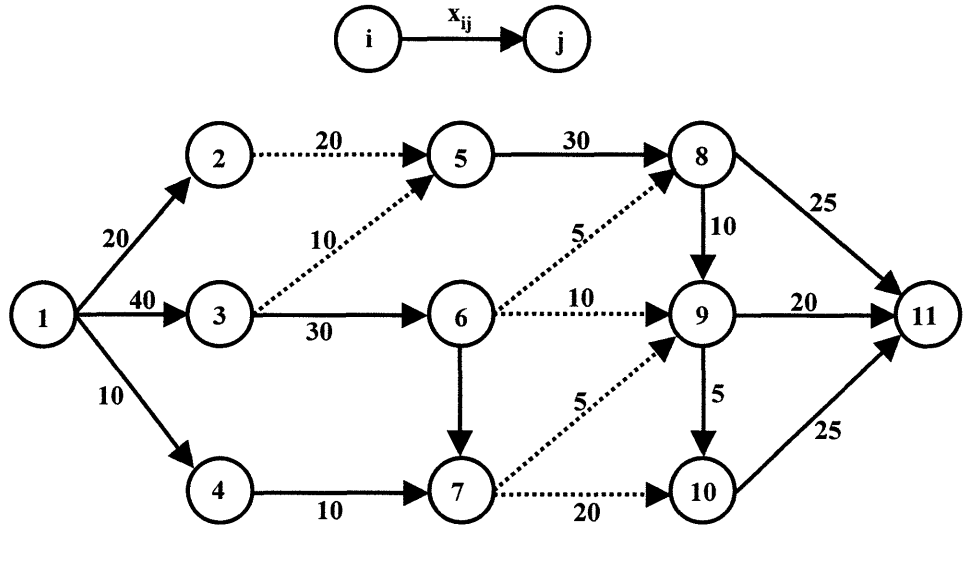
To summarize, we have shown that the inverse minimum cut problem reduces to solving a minimum cut problem that can be solved using any maximum flow algorithm. Currently, the fastest strongly polynomial bound to solve the minimum cut problem (and the maximum flow problem) is $O(nm \log(n^2/m))$ and is 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 $\mathbf{U} = \max\{u_{ij} : (i, j) \in A\}$.

Weighted Case

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:



(a)



(b)

Figure 3. Solution of an inverse minimum cut problem.

$$\text{Minimize } \sum_{(i,j)\in A} u_{ij}\, y_{ij} \tag{5.5a}$$

subject to

$$\sum_{(i,j)\in P} y_{ij} \geq 1, \qquad \text{for all } P \in \mathbf{C}(G'), \tag{5.5b}$$

$$0 \leq y_{ij} \leq w_{ij}, \qquad \text{for all } (i, j) \text{ such that } y^0_{ij} = 0, \tag{5.5c}$$

$$1\text{-}w_{ij} \leq y_{ij} \leq 1, \qquad \text{for all } (i, j) \text{ such that } y^0_{ij} = 1. \tag{5.5d}$$

It can be shown that the dual of (5.5) is a minimum cost flow problem. Hence the weighted inverse minimum cut problem can be solved using any minimum cost flow algorithm.

6. THE INVERSE MINIMUM COST FLOW PROBLEM

In this section, we study the inverse minimum cost flow problem. The weighted inverse minimum cost flow problem has earlier been studied by Huang and Liu [1995] and Sokkalingam [1996], and they show that it reduces to a minimum cost flow problem. We obtain the same result, and in addition we show that the unit weight inverse minimum cost flow problem reduces to a unit capacity minimum cost flow problem and thus can be solved more efficiently.

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:

$$\text{Minimize } \sum_{(i,j)\in A} c_{ij}\, x_{ij} \tag{6.1a}$$

subject to

$$\sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = b(i), \qquad \text{for all } i \in N, \tag{6.1b}$$

$$0 \leq x_{ij} \leq u_{ij}, \qquad \text{for all } (i, j) \in A. \tag{6.1c}$$

In the inverse minimum cost flow problem, we are given a feasible solution x^0 of (6.1) 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 Theorem 1 that the 0-centered dual inverse problem of (6.1) is the following linear programming problem:

$$\text{Minimize } \sum_{(i,j)\in A} c_{ij}\, y_{ij} \tag{6.2a}$$

subject to

$$\sum_{\{j:(i,j)\in A\}} y_{ij} - \sum_{\{j:(j,i)\in A\}} y_{ji} = 0, \quad \text{for all } i \in N, \tag{6.2b}$$

$$y_{ij} \geq 0, \quad \text{for all } (i, j) \in L, \tag{6.2c}$$

$$y_{ij} \leq 0, \quad \text{for all } (i, j) \in U, \tag{6.2d}$$

$$-1 \leq y_{ij} \leq 1, \quad \text{for all } (i, j) \in A. \tag{6.2e}$$

The constraints (6.2c), (6.2d), and (6.2e) can alternatively be stated as follows:

$$0 \leq y_{ij} \leq 1, \quad \text{for all } (i, j) \in L, \tag{6.3a}$$

$$-1 \leq y_{ij} \leq 1, \quad \text{for all } (i, j) \in F, \tag{6.3b}$$

$$-1 \leq y_{ij} \leq 0, \quad \text{for all } (i, j) \in U. \tag{6.3c}$$

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:

$$\text{Minimize} \sum_{(i,j)\in A(x^0)} c'_{ij}\, y'_{ij} \tag{6.4a}$$

subject to

$$\sum_{\{j:(i,j)\in A(x^0)\}} y'_{ij} - \sum_{\{j:(j,i)\in A(x^0)\}} y'_{ji} = 0, \quad \text{for all } i \in N, \tag{6.4b}$$

$$0 \le y'_{ij} \le 1, \qquad \text{for all } (i, j) \in A(x^0). \tag{6.4c}$$

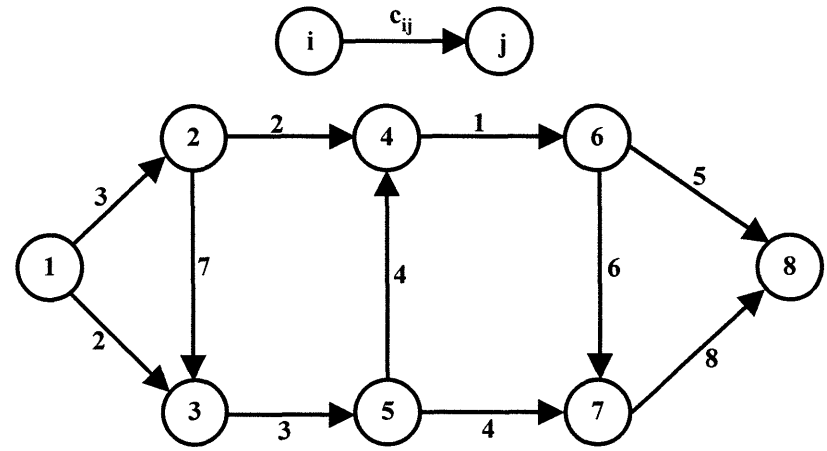
Now observe that (6.4) is the formulation of the *minimum cost circulation problem* (that is, the minimum cost flow problem with zero supply/demand vector) on a unit capacity network. Further, the network on which the minimum cost circulation problem is solved is known as the residual network of G corresponding to the flow x^0 where all arc residual capacities are set to one.

The minimum cost flow problem (6.4) is in general easier to solve than the original minimum cost flow problem (6.1) 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, this minimum cost circulation problem can be solved in $O(O(\min\{n^{5/3}, m^{3/2}\}\log(n\mathbf{C}))$ time, using the algorithm due to Gabow and Tarjan [1989], where $\mathbf{C} = \max\{|c_{ij}| : (i, j) \in A\}$.

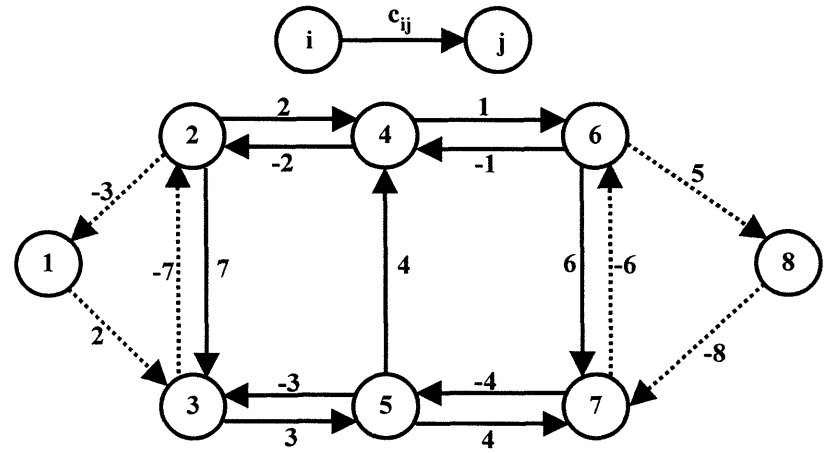
We now explain how to obtain the optimal cost vector d^*. Let π denote the optimal dual variables associated with (6.4b), and $c^{\pi}_{ij} = c_{ij} - \pi_i + \pi_j$ denote the optimal reduced costs. It follows from our discussion in Section 2 that the optimal cost vector d^* is given by

$$d^*_j = \begin{cases} c_j - |c^{\pi}_j| & \text{if } c^{\pi}_j > 0 \text{ and } x^0_j > 0, \\ c_j + |c^{\pi}_j| & \text{if } c^{\pi}_j < 0 \text{ and } x^0_j < u_{ij}, \\ c_j & \text{otherwise.} \end{cases} \tag{6.5}$$

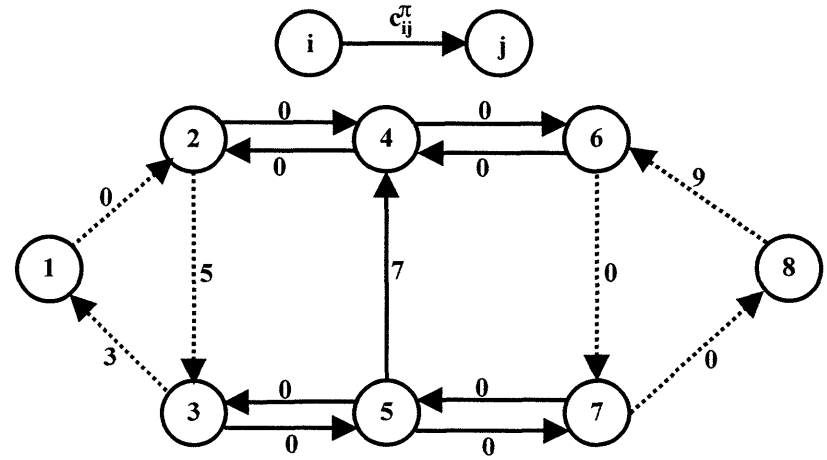
We illustrate our algorithm for the inverse minimum cost flow problem using a numerical example. Figure 4(a) shows a minimum cost flow problem where arc costs are as shown and each arc capacity equals 2. The flow x^0 consists of sending 1 unit of flow along each of the two paths: 1-2-4-6-7-8 and 1-2-3-5-7-8. Figure 4(b) shows the residual



(a)



(b)



(c)

Figure 4. Solution of an inverse minimum cost flow problem.

network $G(x^0)$. Each arc in $G(x^0)$ has unit capacity. The optimal circulation in $G(x^0)$ consists of sending unit flows along each of the cycles 1-3-2-1 and 6-8-7-6. Figure 4(c) optimal reduced costs of arcs. The solution y^* is as follows: $y^*_{12} = -1$, $y^*_{23} = -1$, $y^*_{13} = 1$, $y^*_{68} = 1$, $y^*_{78} = -1$, and $y^*_{67} = -1$. Applying (6.5) yields that we must increase the costs of arcs (1, 3) and (6, 8) by 3 and 9 units, and decrease the cost of arc (2, 3) by 5 units.

Weighted Case

For the weighted version of the inverse minimum cost flow problem, we get the same formulation as (6.4) except that the constraints (6.4c) replaced by the following constraint:

$$0 \le y'_{ij} \le w_{ij}, \qquad \text{for all } (i, j) \in A(x^0). \qquad (6.4c')$$

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.

7. THE MINIMAX INVERSE MINIMUM COST FLOW PROBLEM

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 (6.1) which we wish to make optimal by perturbing the arc costs in a manner so that the maximum perturbation is minimum. 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 Theorem 2 that the 0-centered minimax dual inverse problem of (6.1) is the following linear programming problem in the residual network $G(x^0)$:

$$\text{minimize } \sum_{(i,j)\in A} c_{ij}\, y_{ij} \qquad (7.1a)$$

subject to

$$\sum_{\{j:(i,j)\in A\}} y_{ij} - \sum_{\{j:(j,i)\in A\}} y_{ji} = 0, \quad \text{for all } i \in N, \tag{7.1b}$$

$$y_{ij} \geq 0, \quad \text{for all } (i, j) \in L, \tag{7.1c}$$

$$y_{ij} \leq 0, \quad \text{for all } (i, j) \in U, \tag{7.1d}$$

$$\Sigma_{(i,j)\in A} |y_{ij}| \leq 1. \tag{7.1e}$$

We now perform the same transformation of variables as we did in Section 6 to transform (7.1) to a standard minimum cost flow problem 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:

$$\text{Minimize} \quad \sum_{(i,j)\in A(x^0)} c'_{ij}\, y'_{ij} \tag{7.2a}$$

subject to

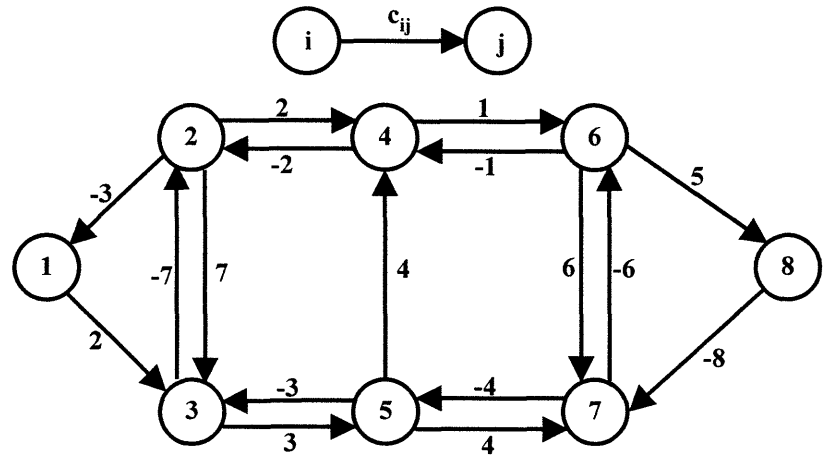
$$\sum_{\{j:(i,j)\in A(x^0)\}} y'_{ij} - \sum_{\{j:(j,i)\in A(x^0)\}} y'_{ji} = 0 \quad \text{for all } i \in N, \tag{7.2b}$$

$$\Sigma_{(i,j)\in A(x^0)}\, y'_{ij} \leq 1, \tag{7.2c}$$

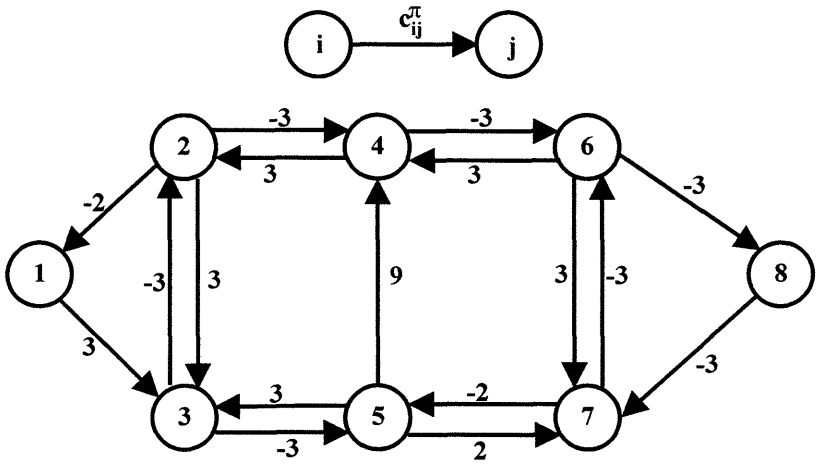
$$y'_{ij} \geq 0, \quad \text{for all } (i, j) \in A(x^0). \tag{7.2d}$$

It is well known that (7.2) is the formulation of the minimum mean cycle problem. A *minimum mean cycle* in the residual network $G(x^0)$ is a directed cycle W for which the mean cost given by $\Sigma_{(i,j)\in W} c_{ij} /|W|$ 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 $O(\sqrt{n}\, m \log \mathbf{C})$ 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 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).

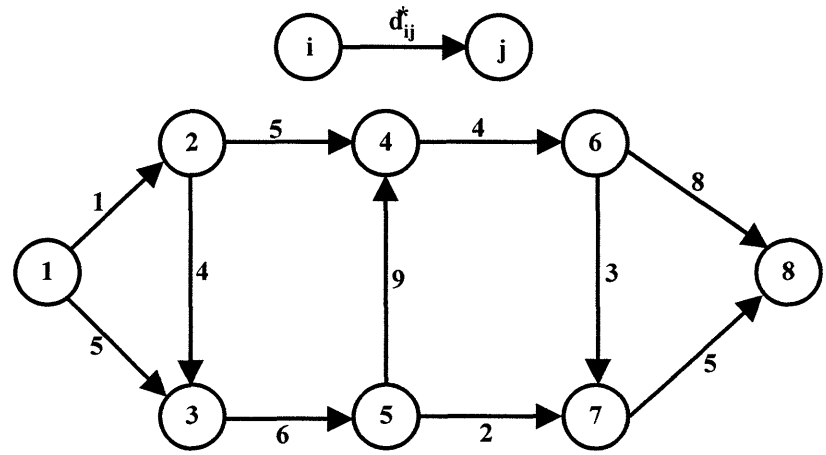
We illustrate our algorithm for the inverse minimum cost flow problem using a numerical example shown in Figure 4(a), where arc costs are as shown and each arc capacity equals 2. The flow x^0 consists of sending 1 unit of flow along each of the two paths: 1-2-4-6-7-8 and 1-2-3-5-7-8. Figure 5(a) shows the residual network $G(x^0)$. The minimum mean cycle in $G(x^0)$ has mean cost -3, and the optimal dual variables are given



(a)



(b)



(c)

Figure 5. Solution of a minimax inverse minimum cost flow problem.

by {0, -1, -5, -6, -11, -10, -13, -18}. With respect to these dual variables, the optimal reduced costs are shown in Figure 5(b). The optimal cost vector d^* computed using (6.5) is shown in Figure 5(c). It can be verified that with respect to the modified arc costs, every directed path from node 1 to node 8 has cost 18.

Weighted Case

For the weighted case, we get the same formulation as in (7.2) except that (7.2c) is replaced by $\Sigma_{\{(i,j)\in G(x^0):\ w_{ij}\neq 0\}} (y_{ij}/w_{ij}) \leq 1$ and $y_{ij} = 0$ if $w_{ij} = 0$, which is the formulation of the minimum cost-to-weight ratio cycle problem (also known as the *tramp steamer problem*). The minimum cost-to-weight ratio problem is to identify a directed cycle W in the network for which $(\Sigma_{(i,j)\in W} c_{ij})/ (\Sigma_{(i,j)\in W} w_{ij})$ is minimum. Using an algorithm due to Lawler [1966], the minimum cost-to-weight ratio problem can be solved in O(nm log(**CW**)) time using Lawler's 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 $\mathbf{W} = \{w_{ij} : (i, j) \in A\}$, 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.

ACKNOWLEDGEMENTS

We gratefully acknowledge the support from the Office of Naval Research under contract ONR N00014-96-1-0051 as well as a grant from the United Parcel Service. We 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, and J. B. Orlin. 1993. *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, NJ.

Ahuja, R. K., A. V. Goldberg, J. B. Orlin, and R. E. Tarjan. 1992. Finding minimum cost flows by double scaling. *Mathematical Programming* **53**, 243-266.

Ahuja, R. K., and J. B. Orlin. 1998a. Inverse Optimization, Part 1: Linear programming and general problem. Working Paper, Sloan School of Management, MIT, Cambridge, MA.

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., B. Pulleyblank, and Ph. L. Toint. 1997. The inverse shortest paths problem with upper bounds on shortest paths costs. In *Network Optimization*, edited by P. Pardalos, D. W. Hearn, and W. H. Hager, *Lecture Notes in Economics and Mathematical Systems*, Volume 450, pp. 156-171.

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.

Cai, M. and X. Yang. 1994. Inverse shortest path problems. Technical Report, Institute of Systems Sciences, Academia Sinica, Beijing, China.

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.

Dial, B. 1996. 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.

Goldberg, A. V., and R. E. Tarjan. 1987. Solving minimum cost flow problem by successive approximation. *Proceedings of the 19th ACM Symposium on the theory of Computing*, pp. 7-18. Full paper in *Mathematics of Operations Research* **15**(1990), 430-466.

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

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

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.

Huang, S., and Z. Liu. 1995. On the inverse version of the minimum cost flow problem. Working Paper, Dept. of ISMT, School of Business and Management, Hong Kong University of Science and Technology, Hong Kong.

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

Orlin, J. B. 1988. A faster strongly polynomial minimum cost flow algorithm. *Proceedings of the 20th ACM Symposium on the Theory of Computing*, pp. 377-387.

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

Sokkalingam, P.T., 1996. *The Minimum Cost Flow Problem : Primal Algorithms and Cost Perturbations*. Unpublished Dissertation, Department of Mathematics, Indian Institute of Technology, Kanpur, INDIA.

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., and J. Zhang. 1996. Inverse maximum capacity path with upper bound contraints. To appear in *OR Spektrum*.

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

Zhang, J., and M. C. Cai. 1998. Inverse problem of minimum cuts. *Mathematical Methods of Operations Research* **47**, No. 1.

Zhang, J., Z. Ma, and C. Yang. 1995. A column generation method for inverse shortest path problems, *ZOR-Mathematical Methods for Operations Research* **41**, 347-358.