PERGAMON

Applied Mathematics Letters 12 (1999) 69–72

Applied Mathematics Letters

Two General Methods for Inverse Optimization Problems

C. YANG AND J. ZHANG
Department of Mathematics
City University of Hong Kong, Hong Kong

(Received and accepted December 1997)

Abstract—We formulate a group of inverse optimization problems as a uniform LP model and provide two computation methods. One is a column generation method which generates necessary columns for simplex method by solving the original optimization problem. Another is an application of the ellipsoid method which can solve the group of inverse problems in polynomial time provided that the original problem has a polynomial-order algorithm. © 1998 Elsevier Science Ltd. All rights reserved.

Keywords—Revised simplex method, Column generation method, Ellipsoid method.

1. INTRODUCTION

Recently, some inverse problems such as inverse shortest path and inverse minimum spanning tree problems have been discussed (see [1–5]). In fact, the concept of inverse problems can be generalized to other network or combinatorial optimization problems. Suppose solutions of an optimization model depend on a set of parameters (e.g., weights, costs, lengths, or capacities), denoted by $\tilde{c}$. Due to frequent use of the model, knowing the values of these parameters quite accurately is important. However, we often know only their estimated values c. If the optimal solutions of the model under certain circumstances are known by experiments or by experience, then we can adjust the parameter vector c by using this solution information. This process is just an inverse optimization problem.

To describe the problem more generally, suppose $G = (V, E, c)$ is a given network consisting of vertices $V = \{v_1, \dots, v_n\}$ and edges $E = \{e_1, \dots, e_m\}$, and each edge e_j is assigned a weight c_j. Let $S^i = \{S^{i1}, \dots, S^{ik_i}\}$ $(i = 1, 2, \dots, r)$, in which each S^{ij}, $j = 1, 2, \dots, k_i$, is a subset of E. For each $S^{ij} \subset E$, let $A^{ij} = (A_1^{ij}, \dots, A_m^{ij})^{\mathrm{T}}$ be its incidence vector defined by setting $A_k^{ij} = 1$ if $e_k \in S^{ij}$ and $A_k^{ij} = 0$, otherwise, for $k = 1, \dots, m$, then the original optimization problem can be written as for each $i = 1, \dots, r$, solve

$$(\mathrm{P}_i) \qquad \begin{aligned} \min \quad & c^{\mathrm{T}} A^{ij}, \\ \text{s.t.} \quad & S^{ij} \in S^i. \end{aligned}$$

The authors gratefully acknowledge the partial support of the Universities Grant Council of Hong Kong (City University Grant 9040189).

We now state its inverse problem. Suppose $S^{i*} \in S^i$, $i = 1, 2, \ldots, r$ are r given subsets of E, and the costs for increasing or decreasing c_i by one unit are, respectively, ℓ_i and d_i. The inverse problem is to minimize the total cost of adjusting the parameter vector c such that all given S^{i*} become the optimal element in S^i $(i = 1, 2, \ldots, r)$. For example, if (v_{s_i}, v_{t_i}), $i = 1, 2, \ldots, r$, are r pairs of vertices, S^i is the set of all simple paths from v_{s_i} to v_{t_i}, and S^{i*} is a specified simple path between v_{s_i} and v_{t_i}, then the problem becomes the inverse shortest path problem; and if $r = 1$ and S^1 consists of all spanning trees of a graph, then it is an inverse minimum spanning tree problem.

Let $u = (u_1, \ldots, u_m)$ and $w = (w_1, \ldots, w_m)$ be, respectively, the increment and decrement of c such that the adjusted parameter vector is $c + u - w$, then the inverse problem is

$$
\begin{aligned}
\text{(INVP)} \qquad & \min && \sum_{k=1}^{m} \ell_k u_k + \sum_{k=1}^{m} d_k w_k, \\
& \text{s.t.} && (c + u - w)^{\mathrm{T}} A^{ij} \geq (c + u - w)^{\mathrm{T}} A^{i*}, && (1.1) \\
& && i = 1, 2, \ldots, r, \quad j = 1, 2, \ldots, k_i, \\
& && 0 \leq u \leq \alpha, \qquad 0 \leq w \leq \beta, && (1.2)
\end{aligned}
$$

where α and β are two constant vectors giving upper bounds for the adjustments of c.

In this paper, we shall propose two general methods for solving this type of inverse optimization problems. The first one, presented in Section 2, is a column generation method which is quite efficient computationally, and the second one, given in Section 3, is an ellipsoid method which can be a polynomial method. In using both methods, we assume reasonably that the original problems (P_i) are solvable. In what follows, a vector may appear as either a row vector, or a column vector, depending on which way is more convenient.

2. COLUMN GENERATION METHOD

If we introduce a dual variable y_{ij} for each constraint (1.1), dual variables $y^1 = (y_1, \ldots, y_m)$ and $y^2 = (y_{m+1}, \ldots, y_{2m})$ for the two groups of constraints in (1.2), then the dual problem of (INVP) is

$$
\begin{aligned}
\text{(DP1)} \qquad & \max && \sum_{i,j} c^{\mathrm{T}} \left(A^{i*} - A^{ij}\right) y_{ij} - \alpha^{\mathrm{T}} y^1 - \beta^{\mathrm{T}} y^2, \\
& \text{s.t.} && \sum_{i,j} y_{ij} \left(A^{ij} - A^{i*}\right) - y^1 \leq \ell, \\
& && -\sum_{i,j} y_{ij} \left(A^{ij} - A^{i*}\right) - y^2 \leq d, \\
& && y^1 \geq 0, \; y^2 \geq 0, \text{ and } y_{ij} \geq 0, \quad \forall i, \forall j.
\end{aligned}
$$

By introducing slack variables $y^3 = (y_{2m+1}, \ldots, y_{3m})$ and $y^4 = (y_{3m+1}, \ldots, y_{4m})$, and setting $y = (y^1, y^2, y^3, y^4)$, $(a_1, \ldots, a_{2m}) = -I_{2m}$, $(a_{2m+1}, \ldots, a_{4m}) = I_{2m}$, and

$$
(\bar{c}_1, \ldots, \bar{c}_m) = (-\alpha_1, \ldots, -\alpha_m), \qquad (\bar{c}_{m+1}, \ldots, \bar{c}_{2m}) = (-\beta_1, \ldots, -\beta_m), \qquad \bar{c}_j = 0, \quad 2m + 1 \leq j \leq 4m,
$$

$$
\bar{c}_{ij} = c^{\mathrm{T}}\left(A^{i*} - A^{ij}\right), \qquad a_{ij} = \begin{pmatrix} A^{ij} - A^{i*} \\ A^{i*} - A^{ij} \end{pmatrix}, \qquad \forall i, \forall j,
$$

problem (DP1) then can be expressed simply as

$$\text{(DP2)} \qquad \begin{aligned} \max \quad & \sum_{i,j} \bar{c}_{ij} y_{ij} + \sum_{j=1}^{4m} \bar{c}_j y_j, \\ \text{s.t.} \quad & \sum_{i,j} a_{ij} y_{ij} + \sum_{j=1}^{4m} a_j y_j = (\ell, d)^{\mathsf{T}}, \\ & y \geq 0, \text{ and } y_{ij} \geq 0, \qquad \forall i, \forall j. \end{aligned}$$

We now consider a column generation method which starts with choosing $2m$ columns to form a feasible basis in problem (DP2), and other columns are not required to know. In fact, we may choose the initial basis with the basic variables $y_B = (y^3, y^4)$ and $\bar{c}_B = 0_{2m}$. The associated basic feasible solution is $y^1 = y^2 = 0$, $y^3 = w$, $y^4 = d$, and $y_{ij} = 0$ for any i and j. Its associated dual variables are $\pi = (\pi^1, \pi^2) = \bar{c}_B^{\mathsf{T}} B^{-1} = 0$. Let z be the corresponding reduced cost coefficients, then

$$z_j = \pi a_j - \bar{c}_j = \alpha_j \geq 0, \quad (j = 1, \ldots, m), \qquad z_j = \pi a_j - \bar{c}_j = \beta_j \geq 0, \quad (j = m+1, \ldots, 2m),$$
$$z_j = 0, \quad (j = 2m+1, \ldots, 4m), \qquad \text{and} \qquad z_{ij} = \pi a_{ij} - \bar{c}_{ij} = c^{\mathsf{T}} \left(A^{ij} - A^{i*}\right).$$

Clearly, this basis is optimal if and only if all $z_{ij} \geq 0$, i.e.,

$$c^{\mathsf{T}} A^{ij} \geq c^{\mathsf{T}} A^{i*},$$

in other words, each S^{i*} is the optimal solution of problem (P$_i$), $i = 1, 2, \ldots, r$. These conditions can be checked by solving problems (P$_i$) for $i = 1, 2, \ldots, r$.

Generally, suppose the current basis $B = (a_{j_1}, a_{j_2}, \ldots, a_{j_{2m}})$, $\pi = \bar{c}_B^{\mathsf{T}} B^{-1} = (\pi^1, \pi^2)$, we can use the same method to check optimality and to decide a pivot column when we have to continue computation. In particular, the reduced cost coefficient of y_{ij} can be calculated by the following formula:

$$\begin{aligned} z_{ij} = \pi a_{ij} - \bar{c}_{ij} &= (\pi^1, \pi^2) \begin{pmatrix} A^{ij} - A^{i*} \\ A^{i*} - A^{ij} \end{pmatrix} - c^{\mathsf{T}} \left(A^{i*} - A^{ij}\right) \\ &= \left(c + \pi^1 - \pi^2\right)^{\mathsf{T}} A^{ij} - \left(c + \pi^1 - \pi^2\right)^{\mathsf{T}} A^{i*}. \end{aligned}$$

Let $\tilde{c} = c + \pi^1 - \pi^2$, then $z_{ij} \geq 0$ if and only if

$$\tilde{c}^{\mathsf{T}} A^{ij} \geq \tilde{c}^{\mathsf{T}} A^{i*}. \tag{2.1}$$

Inequality (2.1) may contain a huge number of constraints, but we can check all of them by solving problem (P$_i$) under the weight vector $\tilde{c}$. If S^{i*} is not an optimal solution, a $z_{i_0 j_0} < 0$ as well as the corresponding element $S^{i_0 j_0}$ in the set S^{i_0} should be found. Then the column

$$a_{i_0 j_0} = \begin{pmatrix} A^{i_0 j_0} - A^{i_0 *} \\ A^{i_0 *} - A^{i_0 j_0} \end{pmatrix}, \tag{2.2}$$

$$\bar{c}_{i_0 j_0} = c^{\mathsf{T}} \left(A^{i_0 *} - A^{i_0 j_0}\right) \tag{2.3}$$

should be brought into the basis to replace a column there, and we can employ the revised simplex method to obtain an improved basic feasible solution by only one extra pivot. Obviously, this method is finitely convergent if the algorithm for solving the original problem is so.

3. ELLIPSOID METHOD

The main advantage of the above method is that it does not need to list all constraints which may expand in an exponential order when m and n increase, and a violated constraint in the inverse problem can be identified by solving the original problem. We find that a suitable version of ellipsoid method can also well fit the inverse problem and share this feature. Furthermore, it provides a polynomial order algorithm for a large group of inverse problems.

Grotschel *et al.* [6] have applied the ellipsoid method to combinatorial optimization problems. They considered the problem

$$\text{(CP)} \qquad \max \ \rho^{\mathrm{T}} x, \qquad x \in P \subset R^n,$$

and constructed a sequence of ellipsoids $E_0, E_1, \dots, E_k, \dots$. Each ellipsoid E_k can be expressed as

$$E_k = \left\{ x \in R^n \mid (x - x_k)^{\mathrm{T}} B_k^{-1} (x - x_k) \le 1 \right\},$$

where x_k is the centre and B_k is a positive definite symmetric matrix. The basic operation of their method can be divided between two routines. The master, or optimization routine (OPT) performs the calculations associated with updating x_k and B_k, and testing for termination. It then calls (with $z = x_k$) a separation routine (SEP) which solves the following problem.

SEPARATION PROBLEM. Given a vector $z \in R^n$, decide whether $z \in P$ or not, and if not, find a hyperplane that separates z from P, i.e., find a vector $\nu \in R^n$ so that $\nu z > \nu y$, $\forall y \in P$.

The separation routine supplies the optimization routine either with the information that $x_k \in P$, or with the required vector ν. In the former case, an outward normal to the next cut is set to $-\rho$ and in the latter case, it is set to ν. The optimization routine can then calculate x_{k+1} and B_{k+1} with some very simple formulae, see [7, equations (2.5),(2.6)].

If in problem (INVP) we express variables (u, w) and parameter vector (ℓ, d) by x and $-\rho$, respectively, and replace $2m$ by n and the constraints (1.1),(1.2) by $x \in P$, then it just becomes problem (CP). We can use the ellipsoid method to solve the inverse problem. In particular, now the separation problem is given $(u, w) \in R^{2m}$, determine if (u, w) is a feasible solution of (INVP), and if not, find a violated inequality. The job of checking whether the constraints in (1.1) are met can again be fulfilled by solving the original problems (P_i) for each $i = 1, 2, \dots, r$ under the parameter vector $c + u - w$. Note that if problems (P_i) can be solved in polynomial time within a common upper bound $P(m, n)$, then we can check the constraints (1.1) in $rP(m, n)$ time. Therefore, if r subjects to a given polynomial upper bound $P'(m, n)$, then the separation problem can be solved in polynomial time $P'(m, n) \cdot P(m, n)$. It is a well-known result that the optimization routine of the ellipsoid method need to call the separation routine for only a polynomial number of times with respect to the size of the LP problem and the length L of the input (see [6,7]). So, we have the following.

THEOREM. *When the ellipsoid method is applied, the inverse optimization problem (INVP) can be solved in polynomial time with respect to m, n, and L, provided that a polynomial order algorithm is available for the original problems* (P_i) *and r has a polynomial upper bound.*

REFERENCES

1. D. Burton and Ph.L. Toint, On an instance of the inverse shortest paths problem, *Mathematical Programming* **53**, 45–61, (1992).
2. S. Xu and J. Zhang, An inverse problem of the weighted shortest path problem, *Japan Journal of Industrial and Applied Mathematics* **12**, 47–60, (1995).
3. J. Zhang, Z. Liu and Z. Ma, On the inverse problem of minimum spanning tree with partition constraints, *Mathematical Methods of Operations Research* **44**, 171–188, (1996).
4. J. Zhang and Z. Liu, Calculating some inverse linear programming problems, *Journal of Computational and Applied Mathematics* **72**, 261–273, (1996).