

THE INVERSE OPTIMAL VALUE PROBLEM*

SHABBIR AHMED[†] AND YONGPEI GUAN

ABSTRACT. This paper considers the following inverse optimization problem: given a linear program, a desired optimal objective value, and a set of feasible cost coefficients, determine a cost-coefficient vector such that the corresponding optimal objective value of the linear program is closest to the given value. The above problem, referred here as the inverse optimal value problem, is significantly different from standard inverse optimization problems that involve determining a cost coefficient vector for a linear program such that a pre-specified *solution vector* is optimal. In this paper, we show that the inverse optimal value problem is NP-hard in general. We identify conditions under which the problem reduces to a concave maximization or a concave minimization problem. We provide sufficient conditions under which the associated concave minimization problem and, correspondingly, the inverse optimal value problem is polynomially solvable. For the case when the set of feasible cost coefficients is polyhedral, we describe an algorithm for the inverse optimal value problem based on solving linear and bilinear programming problems. Some preliminary computational experience is reported.

KEYWORDS. Inverse optimization, Complexity, Linear programming, Bilinear programming.

1. INTRODUCTION

Inverse optimization consists of inferring the parameters – such as objective function and constraint coefficients – of an optimization problem from a pre-specified optimal solution to the problem. A standard inverse optimization problem that has often been studied is as follows: given an optimization problem with a linear objective $P : \min_x \{c^T x \mid x \in X\}$ and a desired optimal solution $x^* \in X$, find a cost vector c^* such that x^* is an optimal solution of P . Typically, c^* is required to satisfy some additional conditions, for example, given a preferred cost coefficient vector \hat{c} , the deviation $\|c^* - \hat{c}\|_p$ is to be minimum under some ℓ_p -norm. The above class of problems were introduced by Burton and Toint [7, 8] in the context of identifying edge lengths to induce a set of pre-specified paths to be the set of shortest paths in a graph. The authors proposed a convex quadratic programming approach for a model in which the deviation is measured in the ℓ_2 -norm. When the underlying optimization problem P is a general linear program, Zhang and Liu [18, 19] discussed linear programming approaches for the ℓ_1 and ℓ_∞ case. Recently, Ahuja and Orlin [1] proved that if the underlying optimization problem P is polynomially solvable, then the standard inverse optimization problem under the ℓ_1 or ℓ_∞ norm is also polynomially solvable. For a comprehensive survey of the literature on inverse optimization, the reader is referred to Heuberger [10].

* This research has been supported in part by the National Science Foundation under CAREER Award DMI-0133943.

[†] Corresponding author, E-mail: sahmed@isye.gatech.edu.

In this paper, we consider the following generalization of the above standard inverse optimization problem: given the optimization problem P and a desired optimal objective value z^* , determine a cost-coefficient vector c^* such that the corresponding optimal objective value of P is closest to z^* . Note that here the *optimal objective value* rather than the *optimal solution* is specified. We refer to this problem as the *inverse optimal value problem* to distinguish it from standard inverse optimization problems.

Unlike standard inverse optimization, the inverse optimal value problem has received little attention in the literature. One of the earliest works that allude to this class of problems is a minimax (center) location model due to Berman *et al.* [5]. Here the authors considered a problem of determining edge lengths in a graph such that the induced minimax distance from a given vertex to all other vertices is within prescribed bounds. The authors showed that the problem is NP-complete for general graphs and described a mixed-integer programming formulation. For tree graphs, the authors described a linear programming formulation. Zhang *et al.* [20] suggested a strongly polynomial algorithm for the above inverse location problem for tree graphs. Burton *et al.* [6] considered a generalization of the standard inverse optimization problems described in [7, 8]. Here, instead of exactly specifying the shortest paths, desired upper bounds on the shortest path lengths are specified. The goal is to identify edge lengths in the graph such that the induced shortest path lengths satisfy the pre-specified upper bounds. The authors proved that obtaining a global solution to this problem is NP-complete, and provided an algorithm for obtaining local solutions. Fekete *et al.* [9] discussed a similar problem of determining edge lengths such that these lengths exactly induce pre-specified shortest path lengths between vertex pairs. The authors proved that the problem is NP-complete, and identified some polynomially solvable cases. Paleologo and Takriti [13] discussed an application of an inverse shortest path length problem similar to that in [9] to determine arbitrage-free prices in a bandwidth network. They suggested a mixed-integer programming formulation for the problem. To our knowledge, apart from the above mentioned context of shortest paths on graphs, the inverse optimal value problem in a general setting has not been addressed previously.

In this paper, we address a general inverse optimal value problem where the underlying optimization problem is a linear program, and set of cost coefficients is restricted to a convex compact set. We show that the general problem is NP-hard. Under very general assumptions, we provide structural characterization under which the problem reduces to a concave maximization or a concave minimization problem. We provide sufficient conditions under which the associated concave minimization problem and, correspondingly, the inverse optimal value problem is polynomially solvable. For the case when the set of feasible cost coefficients is polyhedral, we describe an algorithm for the problem based on solving linear and bilinear programming problems. Finally, we report on some preliminary computational experience.

2. PROBLEM STATEMENT, NOTATION, AND ASSUMPTIONS

In this section, we formally state the inverse optimal value problem under study, introduce some notation, and state the key assumptions used throughout the paper.

Consider the optimal value function of linear program in terms of its cost coefficient vector

$$(1) \quad Q(c) := \min_x \{c^T x \mid Ax = b, x \geq 0\},$$

where $x \in \mathbb{R}^n$. Given a convex set $C \subseteq \mathbb{R}^n$ of the objective cost coefficients and a real number z^* , this paper is concerned with the inverse optimization problem of finding a cost coefficient vector from the set C such that the optimal objective value of the linear program (1) is “close” to z^* . The problem can be formulated as

$$(2) \quad \min_c \{f(c) \mid c \in C\},$$

where $f(c) := |Q(c) - z^*|$ if $Q(c) \in \mathbb{R}$ and $f(c) := +\infty$ if $Q(c) \in \{-\infty, +\infty\}$. We refer to (2) as the inverse optimal value problem. Note that an instance of (2) is given by specifying the linear programming value function Q , the set of feasible cost vectors C , and the desired optimal objective value z^* . We shall denote such an instance by $P(Q, C, z^*)$.

Let us define the sets $C_{z^*} := \{c \mid Q(c) \geq z^*\}$ and $C_\infty := \{c \mid Q(c) > -\infty\}$. We shall frequently refer to the set $\bar{C} := C \cap C_{z^*}$. Given compact sets C and \bar{C} , let c^L be a point such that $c_j^L = \min\{c_j \mid c \in C\}$ for all $j = 1, \dots, n$, and \bar{c}^L be a point such that $\bar{c}_j^L = \min\{c_j \mid c \in \bar{C}\}$ for all $j = 1, \dots, n$. Similarly, we let c^U be a point such that $c_j^U := \max\{c_j \mid c \in C\}$ for all $j = 1, \dots, n$. Given a point $c \in C$, we let $[c^L, c]$ denote the line-segment joining c^L to c . We denote the “lower boundary” of C as $\partial_L C := \{c \in C \mid [c^L, c] \cap C = \{c\}\}$. It is easily verified that $\partial_L C \subseteq \partial C$, where ∂C denotes the relative boundary of C . We let $\Omega(C)$ denote the set of extreme points of the convex set C . Similar definitions hold for the set \bar{C} . Finally, we define the notion of non-decreasing functions. Given two vectors a and b in \mathbb{R}^n , we write $a < b$ if $a_{j'} < b_{j'}$ for some j' and $a_j \leq b_j$ for all $j = 1, \dots, n$. A function $g : \mathbb{R}^n \mapsto [-\infty, +\infty]$ is *non-decreasing* if for $a, b \in \mathbb{R}^n$ such that $a < b$, we have $g(a) \leq g(b)$.

Throughout the rest of this paper, we make the following assumptions:

- (A1) The feasible region of the linear program $\{x \mid Ax = b, x \geq 0\} \neq \emptyset$.
- (A2) The set of cost coefficient vectors C is non-empty, compact, and convex.
- (A3) $C \cap C_\infty \neq \emptyset$.

By assumption (A1), we have that $Q : \mathbb{R}^n \mapsto [-\infty, +\infty)$. Using strong duality, we can then write

$$(3) \quad Q(c) = \max_\pi \{\pi^T b \mid \pi^T A \leq c\},$$

and also $C_{z^*} = \{c \mid \exists \pi \text{ s.t. } \pi^T A \leq c, \pi^T b \geq z^*\}$ and $C_\infty = \{c \mid \exists \pi \text{ s.t. } \pi^T A \leq c\}$.

The following properties are easily verified.

Proposition 2.1. (i) $Q(\cdot)$ is upper-semi-continuous (u.s.c.) over \mathbb{R}^n . (ii) $Q(\cdot)$ is piece-wise linear and concave over C_∞ . (iii) The sets C_{z^*} and C_∞ are closed and convex.

Furthermore, the non-negativity restriction in the linear program (1) implies that

Proposition 2.2. $Q(\cdot)$ is non-decreasing over \mathbb{R}^n .

Finally, since $f(\cdot)$ is continuous over the non-empty compact set $C \cap C_\infty$, we have that

Proposition 2.3. *The inverse optimal value problem (2) has a finite optimal solution.*

3. COMPLEXITY

In this section, we prove that the inverse optimal value problem (2) is NP-hard. Our complexity proof relies on establishing equivalence between the inverse optimal value problem and the binary integer feasibility problem. Given an integer matrix $A \in \mathbb{Z}^{m \times n}$, and an integer vector $b \in \mathbb{Z}^m$, the binary integer feasibility problem can be stated as follows

Is there a vector $x \in \{0, 1\}^n$ such that $Ax \leq b$?

An instance of the binary integer feasibility problem is specified by the matrix-vector pair (A, b) . We shall denote such an instance by $B(A, b)$.

Lemma 3.1. *Given an instance $B(A, b)$, we can construct an instance $P(\hat{Q}, \hat{C}, \hat{z}^*)$ of the inverse optimal value problem, such that $B(A, b)$ has an answer “yes” if and only if the optimal objective value of $P(\hat{Q}, \hat{C}, \hat{z}^*)$ is zero.*

Proof. Given an instance $B(A, b)$ with $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, let us define the compact polyhedral set

$$\hat{C} := \left\{ (c_1, c_2, c_3)^T \in \mathbb{R}^{3n} \mid \begin{array}{l} c_1 \in \mathbb{R}^n, c_2 \in \mathbb{R}^n, c_3 \in \mathbb{R}^n, \\ Ac_1 \leq b, c_1 = c_2, c_3 = e, \\ 0 \leq c_1 \leq e, 0 \leq c_2 \leq e \end{array} \right\},$$

and the linear programming value function $\hat{Q} : \mathbb{R}^{3n} \mapsto \mathbb{R}$ as:

$$\hat{Q}(c) := \min c_1^T u - c_2^T v + c_3^T v \\ \text{s.t. } u + v \geq e, u \in \mathbb{R}_+^n, v \in \mathbb{R}_+^n,$$

where $e \in \mathbb{R}^n$ is a vector of ones. Finally, letting $\hat{z}^* = 0$, we have an instance $P(\hat{Q}, \hat{C}, \hat{z}^*)$ of the inverse optimal value problem.

Suppose $B(A, b)$ has an answer “yes,” i.e. there exists $\hat{x} \in \{0, 1\}^n$ such that $A\hat{x} \leq b$. Consider a cost vector $\hat{c} = (c_1, c_2, c_3)^T$ such that $c_1 = c_2 = \hat{x}$ and $c_3 = e$. Clearly $\hat{c} \in \hat{C}$. Now, note that

$$\hat{Q}(\hat{c}) := \sum_{j=1}^n \left(\min \hat{x}_j u_j + (1 - \hat{x}_j) v_j \right. \\ \left. \text{s.t. } u_j + v_j \geq 1, u_j, v_j \geq 0 \right).$$

Since $\hat{x}_j \in \{0, 1\}$, we have $\hat{x}_j = 0$ implies $u_j = 1$ and $v_j = 0$, and $\hat{x}_j = 1$ implies $u_j = 0$ and $v_j = 1$. Thus $\hat{Q}(\hat{c}) = 0 = \hat{z}^*$ and the optimal objective function in $P(\hat{Q}, \hat{C}, \hat{z}^*)$ is zero.

Now suppose the optimal objective value in $P(\hat{Q}, \hat{C}, \hat{z}^*)$ is zero, i.e. there exists $\bar{c} \in \hat{C}$ such that $\hat{Q}(\bar{c}) = 0$. Let $\bar{c} := (\hat{c}, \hat{c}, e)^T$, where $\hat{c} \in \mathbb{R}^n$. Note that

$$\hat{Q}(\bar{c}) = \sum_{j=1}^n \hat{Q}_j(\hat{c}),$$

where

$$\begin{aligned} \hat{Q}_j(\hat{c}) = \min \quad & \hat{c}_j u_j + (1 - \hat{c}_j) v_j \\ \text{s.t.} \quad & u_j + v_j \geq 1, \quad u_j, v_j \geq 0. \end{aligned}$$

Since $0 \leq \hat{c}_j \leq 1$, the optimal value of the above linear program will satisfy $Q_j(\hat{c}) = \min\{\hat{c}_j, 1 - \hat{c}_j\}$ for all j . Furthermore, $\hat{Q}(\bar{c}) = 0$ implies $\hat{Q}_j(\hat{c}) = 0$ for all j . It then follows that $\hat{c}_j \in \{0, 1\}$ for all j . Then, from the fact that $(\hat{c}, \hat{c}, e)^T \in \hat{C}$, we have that the binary vector $x = \hat{c}$ provides an affirmative answer for $B(A, b)$. \square

Theorem 3.1. *The inverse optimal value problem is NP-hard.*

Proof. Lemma 3.1 shows that we can provide an answer to any binary integer feasibility question by constructing and solving an equivalent instance of the inverse optimal value problem. The claim follows from the fact that the binary integer feasibility problem is NP-complete, and that the construction in Lemma 3.1 is clearly polynomial. \square

4. STRUCTURAL RESULTS

In this section, we describe some structural conditions to reduce the inverse optimal value problem to well-known optimization problems. Our analysis centers on whether the set \bar{C} is empty or non-empty.

Proposition 4.1. *Suppose $\bar{C} = \emptyset$. Let c^* be an optimal solution of*

$$(4) \quad \max\{Q(c) \mid c \in C\},$$

then c^ is an optimal solution of the inverse optimal value problem (2).*

Proof: Since $Q(c)$ is u.s.c. over the non-empty, convex, and compact feasible region C , problem (4) has a well-defined optimal solution. Since $\bar{C} = \emptyset$, it follows that $Q(c) < z^*$ for all $c \in C$. Problem (2) then reduces to (4). \square

Using the dual representation (3) of $Q(c)$, we can state problem (4) in the above proposition as:

$$(5) \quad \begin{aligned} \max_{c, \pi} \quad & b^T \pi, \\ \text{s.t.} \quad & c \in C, \\ & \pi^T A - c \leq 0. \end{aligned}$$

The above problem involves maximizing a linear function over a convex set for which a variety of efficient algorithms exists. If C is polyhedral, problem (5) is simply a linear program.

Proposition 4.2. *Suppose $\bar{C} \neq \emptyset$. Let c^* be an optimal solution to*

$$(6) \quad \min\{Q(c) \mid c \in \bar{C}\},$$

then c^ is an optimal solution of the inverse optimal value problem (2).*

Proof: We first argue that there exists a solution c^* to the inverse optimal value problem (2) such that $c^* \in \bar{C}$. Suppose not. Then there exists $c^* \in C \setminus \bar{C}$ such that $f(c^*) \leq f(c)$ for all $c \in C$. Note that $c^* \in C_\infty$, otherwise it is not an optimal solution. Since $c^* \notin C_{z^*}$, we have $Q(c^*) < z^*$. Now, consider a point $c' \in \bar{C}$. Note that $Q(c') \geq z^* > Q(c^*)$. Define $c^\lambda := \lambda c^* + (1 - \lambda)c'$ for any $\lambda \in (0, 1)$. Since both c^* and c' are in the convex set $C \cap C_\infty$, so is c^λ for any $\lambda \in (0, 1)$. Since $Q(\cdot)$ is concave over $C \cap C_\infty$, we have $Q(c^\lambda) \geq \lambda Q(c^*) + (1 - \lambda)Q(c') > Q(c^*)$ for any $\lambda \in (0, 1)$. From the continuity of $Q(\cdot)$, we can choose λ sufficiently small such that $Q(c^\lambda) \leq z^*$. Thus $f(c^\lambda) = z^* - Q(c^\lambda) < z^* - Q(c^*) = f(c^*)$. Therefore c^* cannot be an optimal solution. Thus, if $\bar{C} \neq \emptyset$ there exists an optimal solution c^* to (2) such that $c^* \in \bar{C}$. Note that $Q(c) \geq z^*$ for all $c \in \bar{C}$, thus problem (2) reduces to problem (6), and the claim follows. \square

Problem (6) in the above proposition amounts to minimizing a concave function over a convex set. Using the primal representation of $Q(c)$ and the dual representation of \bar{C} , it can be easily verified that problem (6) is equivalent to the following problem:

$$(7) \quad \begin{aligned} \min_{c,x,\pi} \quad & c^T x, \\ \text{s.t.} \quad & c \in C, \\ & \pi^T A - c \leq 0, \quad \pi^T b \geq z^* \\ & Ax = b, \quad x \geq 0. \end{aligned}$$

The above problem involves minimizing a bilinear objective function over a convex constraint set. When the constraint set is polyhedral, this class of non-convex programs are known as bilinear programs (cf. [2, 11]). Since the variables c and x are not coupled through any constraints, such bilinear problems are referred to as uncoupled or disjoint. A wide variety of optimization techniques have been proposed for solving disjoint bilinear programming problems. In Section 6, we use one such technique in the context of the inverse optimal value problem.

We conclude this section by exploiting the monotonicity property of $Q(c)$ to show that, in case $\bar{C} \neq \emptyset$, there exists a global solution to the inverse optimal value problem that is an extreme point on the ‘‘lower boundary’’ of \bar{C} . In Section 5, this characterization will suggest sufficient conditions under which the inverse optimal value problem is polynomially solvable.

Lemma 4.1. *Given any point $c^* \in \bar{C}$, there exists $c' \in [\bar{c}^L, c^*]$ such that $c' \in \partial_L \bar{C}$.*

Proof: The claim holds trivially if $c^* = \bar{c}^L$. Let us assume $\bar{c}^L < c^*$. Consider the following optimization problem

$$(8) \quad \min\{\lambda \mid \bar{c}^L + \lambda(c^* - \bar{c}^L) \in \bar{C}, 0 \leq \lambda \leq 1\},$$

and let λ' be its optimal solution. Let us define $c' = \bar{c}^L + \lambda'(c^* - \bar{c}^L)$. Suppose $c' \notin \partial_L \bar{C}$, then there exists $c'' \neq c'$ such that $c'' \in [\bar{c}^L, c'] \cap \bar{C}$. Since $[\bar{c}^L, c'] \subseteq [\bar{c}^L, c^*]$, this implies that there exists $\lambda'' \neq \lambda'$ such that $c'' = \bar{c}^L + \lambda''(c^* - \bar{c}^L) \in \bar{C}$. Note that $c'' < c'$, thus $\bar{c}_j^L + \lambda''(c_{j'}^* - \bar{c}_j^L) \leq \bar{c}_j^L + \lambda'(c_{j'}^* - \bar{c}_j^L)$ for all $j = 1, \dots, n$. Recall that $\bar{c}^L < c^*$, thus there exists j' such that $\bar{c}_{j'}^L < c_{j'}^*$. Since $\lambda'' \neq \lambda'$, this then implies $\lambda'' < \lambda'$. Therefore λ' cannot be an optimal solution of (8). \square

Proposition 4.3. *If $\bar{C} \neq \emptyset$, then there exists an optimal solution c^* of the inverse optimal value problem (2) such that $c^* \in \partial_L \bar{C} \cap \Omega(\bar{C})$.*

Proof: We first argue that the set $\partial_L \bar{C} \cap \Omega(\bar{C})$ is non-empty. Let $S = \operatorname{argmin}\{e^T c \mid c \in \bar{C}\}$. We claim that $S \subseteq \partial_L \bar{C}$. Suppose not. Consider $c' \in S \setminus \partial_L \bar{C}$. Since $c' \in \bar{C}$, by Lemma 4.1, there exists $c'' \in \partial_L \bar{C}$ such that $c'' < c'$, thus $e^T c'' < e^T c'$, and c' cannot be in S . Since $S \cap \Omega(\bar{C}) \neq \emptyset$, we have $\partial_L \bar{C} \cap \Omega(\bar{C}) \neq \emptyset$.

Now note that by Proposition 4.2, problem (2) is equivalent to problem (6). Consider an optimal solution c^* to problem (6) such that $c^* \notin \partial_L \bar{C} \cap \Omega(\bar{C})$. By Lemma 4.1, there exists $c' < c^*$ such that $c' \in \partial_L \bar{C}$. By the non-decreasing property of $Q(\cdot)$, we have $Q(c') \leq Q(c^*)$, therefore c' is also an optimal solution. By convexity of \bar{C} , we can write $c' = \sum_{i \in I} \lambda_i c_i + (1 - \sum_i \lambda_i) c_0$ where I is an appropriate index set, $c_i \in \Omega(\bar{C})$ and $\lambda_i \geq 0$ for $i \in I$, and $c_0 \in \partial_L \bar{C} \cap \Omega(\bar{C})$. Using concavity of Q we have that $Q(c') \geq \sum_i \lambda_i Q(c_i) + (1 - \sum_i \lambda_i) Q(c_0)$. Since $Q(c') \leq Q(c_i)$ for all $i \in I$, we have that $Q(c_0) \leq Q(c')$, thus $c_0 \in \partial_L \bar{C} \cap \Omega(\bar{C})$ is also an optimal solution for the problem. \square

5. CONDITIONS FOR POLYNOMIAL SOLVABILITY

From the analysis of the previous section, it is clear that the difficulty in solving the inverse optimal value problem arises in case $\bar{C} \neq \emptyset$. In this case, we are required to solve the concave minimization problem (6). Recall, that Proposition 4.3 suggests that there exists a global optimal solution to problem (6) that lies on an extreme point on the “lower boundary” of \bar{C} . If we have that $\bar{c}^L \in \bar{C}$ then it is easily verified that $\partial_L \bar{C} \cap \Omega(\bar{C}) = \{\bar{c}^L\}$. Consequently, \bar{c}^L is an optimal solution of (6), and hence of the inverse optimal value problem (2). A more general condition guaranteeing easy solvability of these problems is established next.

Proposition 5.1. *Suppose $\bar{C} \neq \emptyset$ and $\bar{c}^L \in C$. Let c^* be an optimal solution to the following problem*

$$(9) \quad \min\{e^T c \mid c \in \bar{C}\}.$$

Then c^ is an optimal solution to problem (6) and, hence, is an optimal solution to the inverse optimal value problem (2).*

Proof: Suppose not. Then there exists $c' \in \bar{C}$ such that $Q(c') < Q(c^*)$. This implies $Q(c^*) > z^*$. Define $c^\lambda = \bar{c}^L + \lambda(c^* - \bar{c}^L)$. Since $\bar{c}^L, c^* \in C$, so is c^λ for all $\lambda \in (0, 1)$. By the continuity of $Q(\cdot)$ at c^* , we can choose λ sufficiently close to one, such that $c^\lambda \neq c^*$ and $Q(c^\lambda) \geq z^*$. Thus $c^\lambda \in \bar{C}$. Clearly $c^\lambda < c^*$, thus $e^T c^\lambda < e^T c^*$. Therefore c^* cannot be an optimal solution to (9). \square

Problem (9) above is equivalent to

$$(10) \quad \begin{aligned} \min_{c, \pi} \quad & e^T c, \\ \text{s.t.} \quad & c \in C, \\ & \pi^T A - c \leq 0, \quad \pi^T b \geq z^*, \end{aligned}$$

and is a convex program with a linear objective, and can be solved quite efficiently. In particular, when C is polyhedral, problem (10) is simply a linear program.

Theorem 5.1. *If $\bar{c}^L \in C$, and the convex programs (4) and (9) can be solved in polynomial time, then the inverse cost coefficient problem (2) can be solved in polynomial time.*

Proof: If (9) can be solved in polynomial time, then we can verify whether the convex set $\bar{C} = \emptyset$ in polynomial time. If $\bar{C} = \emptyset$, Proposition 4.1 implies that an optimal solution to (2) can be found in polynomial time by solving the convex program (4). If $\bar{C} \neq \emptyset$, then by Proposition 5.1, an optimal solution can be found in polynomial time by solving the convex program (9). \square

Corollary 5.1.1. *If $c^L \in C$, then the inverse optimal value problem is polynomially solvable.*

Proof: It is easily verified that $c^L \in C$ implies that $\bar{c}^L \in C$. \square

We conclude this section by commenting on a particularly easy case of the inverse optimal value problem. Consider the case when both $c^L \in C$ and $c^U \in C$. This is the case, for example when C is only defined by simple upper and lower bounds.

Proposition 5.2. *Suppose $c^L \in C$ and $c^U \in C$. Then the following are true*

- (1) *If $Q(c^L) \geq z^*$ then c^L is an optimal solution of the inverse optimal value problem.*
- (2) *If $Q(c^U) \leq z^*$ then c^U is an optimal solution of the inverse optimal value problem.*
- (3) *If $Q(c^L) < z^* < Q(c^U)$ then there exists an optimal solution $c^* \in [c^L, c^U]$ of the inverse optimal value problem, and $Q(c^*) = z^*$.*

A proof of the above result is trivial and is omitted. The above result suggests that if $c^L \in C$ and $c^U \in C$, then solving the inverse optimal value problem reduces to at most a line search on $[c^L, c^U]$ to find c^* such that $Q(c^*) = z^*$. This line search can be efficiently executed through parametric linear programming as outlined in the next section.

6. SOLVING THE POLYHEDRAL CASE

The analysis in the Section 4 suggests that we can solve the inverse optimal value problem by first checking whether $\bar{C} = \emptyset$, and then solving the corresponding convex problem (5) or the non-convex problem (7). Furthermore, in case $\bar{C} \neq \emptyset$, we can refine this scheme by verifying the condition $\bar{c}^L \in C$, and accordingly solving the convex program (9). In this section, we develop linear programming based procedures to identify and deal with each of the above situations when the set C is polyhedral. In addition to assumptions (A1)-(A3), we shall assume henceforth that

- (A4) The set C is polyhedral, i.e. $C = \{c \mid Bc \leq d\}$.
- (A5) $C \subseteq C_\infty$.

Assumption (A5) guarantees that the underlying linear program is bounded for all cost vectors in C . This assumption allows substantial simplification in solving the non-convex program (7) as detailed later in this section. Note that the assumption is trivially satisfied when the feasible region $\{x \mid Ax = b, x \geq 0\}$ of the underlying LP is bounded.

The Solution Strategy

The first step in our solution procedure is to check if the polyhedral set $\overline{C} = \emptyset$. This can be accomplished by solving problem (10) which, under assumption (A4), reduces to the linear program:

$$(11) \quad \begin{array}{ll} \min_{c,\pi} & e^T c, \\ \text{s.t.} & Bc \leq d, \\ & \pi^T A - c \leq 0, \quad \pi^T b \geq z^*. \end{array}$$

If the above LP is infeasible, we conclude $\overline{C} = \emptyset$. Otherwise, $\overline{C} \neq \emptyset$, and we denote an optimal solution of (11) by c^0 .

If $\overline{C} = \emptyset$, we obtain an optimal solution of the inverse optimal value problem by computing an optimal solution of problem (5) which, under assumption (A4), reduces to the linear program:

$$(12) \quad \begin{array}{ll} \max_{c,\pi} & b^T \pi, \\ \text{s.t.} & Bc \leq d, \\ & \pi^T A - c \leq 0. \end{array}$$

Consider now the case $\overline{C} \neq \emptyset$. We first compute the vector \overline{c}^L by solving the linear program

$$(13) \quad \begin{array}{ll} \overline{c}_j^L = \min_{c,\pi} & e_j^T c, \\ \text{s.t.} & Bc \leq d, \\ & \pi^T A - c \leq 0, \quad \pi^T b \geq z^*. \end{array}$$

for each $j = 1, \dots, n$. Above, e_j is the j -th unit vector. If $\overline{c}^L \in C$, then we conclude that c^0 (the optimal solution of problem (11)) is an optimal solution for the inverse optimal value problem.

If $\overline{c}^L \notin C$, then we need to solve the non-convex program (7), which under assumption (A4), reduces to the disjoint bilinear program:

$$(14) \quad \begin{array}{ll} \min_{c,x,\pi} & c^T x, \\ \text{s.t.} & Bc \leq d, \\ & \pi^T A - c \leq 0, \quad \pi^T b \geq z^* \\ & Ax = b, \quad x \geq 0. \end{array}$$

Note that the above problem is stated in terms of the variables c , x , and π . Assumption (A5) allows for substantial simplification of (14). Recall that problem (14) is equivalent to $\min\{Q(c) \mid c \in \overline{C}\}$. Instead, let us consider the problem

$$(15) \quad \min\{Q(c) \mid c \in C\}.$$

Assumption (A5) guarantees that $Q(c)$ is concave over C , hence problem (15) is well-defined. We can now re-state this problem as

$$(16) \quad \begin{array}{ll} \min_{c,x} & c^T x, \\ \text{s.t.} & Bc \leq d, \\ & Ax = b, \quad x \geq 0. \end{array}$$

Problem (16) avoids inclusion of the π variables in the bilinear formulation and is significantly easier to solve than (14). Note, however, that a solution of (16) is no longer guaranteed to satisfy $Q(c) \geq z^*$, and is, therefore, not necessarily an optimal solution of the inverse optimal value problem. Let (c', x') be a global optimal solution of problem (16). If $c'^T x' \geq z^*$, then clearly $c' \in \overline{C}$ and is, therefore, a global

optimal solution of the inverse optimal value problem. Otherwise if $c'^T x' < z^*$, then we have $Q(c') < z^* \leq Q(c^0)$, where c^0 is an optimal solution of problem (11). By the continuity of $Q(c)$ we know that there exists $c^* \in [c', c^0]$ such that $Q(c^*) = z^*$, hence c^* is an optimal solution of the inverse optimal value problem. Such a c^* is easily computed by parametric linear programming as follows. Let x^0 be an optimal basic solution of the LP corresponding to $Q(c^0)$. Starting from basis of x^0 , solve the parametric linear program

$$(17) \quad F(\lambda) = \min\{(c^0 + \lambda\Delta)^T x \mid Ax = b, x \geq 0\},$$

with $\Delta = c' - c^0$, for $\lambda \in [0, 1]$ to find λ^* such that $F(\lambda^*) = z^*$. Then $c^* = c^0 + \lambda^* \Delta$ is an optimal solution to the inverse optimal value problem.

Algorithm 1 summarizes the above mentioned solution strategy for solving the inverse optimal value problem when the set C is polyhedral.

Algorithm 1 Solution strategy for the inverse optimal value problem (2)

solve the linear program (11).
if problem (11) is infeasible, *i.e.*, $\bar{C} = \emptyset$ **then**
 solve the linear program (12), and let c^* be its optimal solution.
else {problem (11) is feasible, *i.e.*, $\bar{C} \neq \emptyset$ }
 let c^0 be an optimal solution of (11).
 compute \bar{c}^L by solving the linear programs (13) for $j = 1, \dots, n$.
 if $\bar{c}^L \in C$ **then**
 set $c^* \leftarrow c^0$.
 else $\{\bar{c}^L \notin C\}$
 solve the bilinear program (16) and let (c', x') be an optimal solution.
 if $c'^T x' \geq z^*$ **then**
 set $c^* \leftarrow c'$.
 else $\{c'^T x' < z^*\}$
 solve the parametric linear program (17) to find $c^* \in [c', c^0]$ such that
 $Q(c^*) = z^*$.
 end if
 end if
end if
return c^* as the optimal solution.

Disjoint Bilinear Programming

A key step in the solution strategy outlined above is solving the disjoint bilinear program (16). A wide variety of solution strategies have been proposed in the literature – see, for example, [2, 3, 11, 14, 15, 16, 17] and references therein. In this paper we use a particularly simple linear programming based strategy proposed by Bennett and Mangasarian [4]. This scheme starts out with an initial feasible solution to the disjoint bilinear program, and iterates by solving two linear programs – one in terms of the x variables and the other in terms of the c variables– to improve the bilinear objective $c^T x$. The procedure is summarized in Algorithm 2. Note that “arg vertex partial min” denotes an extreme point solution whose objective value is no bigger than that corresponding to the previous iterate. The following result establishes the convergence of the algorithm.

Algorithm 2 Solving the disjoint bilinear program (16)

start with an initial feasible solution (x^0, c^0) .
set $i = 0$.
while there is an improvement **do**
 compute (x^{i+1}, c^{i+1}) from (x^i, c^i) such that
 $x^{i+1} \in \arg \text{vertex partial } \min_x \{c^{i+1T} x \mid Ax = b, x \geq 0\}$,
 $c^{i+1} \in \arg \text{vertex partial } \min_c \{x^{i+1T} c \mid Bc \leq d\}$,
 and $c^{i+1T} x^{i+1} < c^{iT} x^i$.
 set $i \leftarrow i + 1$.
end while
return c^i as the solution.

Proposition 6.1. *Algorithm 2 terminates in a finite number of steps at either a global solution of (16) or a solution (x^{i+1}, c^i) satisfying the necessary optimality condition: $c^{iT}(x - x^{i+1}) + x^{i+1T}(c - c^i) \geq 0$ for all $x \in \{x \mid Ax = b, x \geq 0\}$ and all $c \in \{c \mid Bc \leq d\}$.*

Proof: See [4]. □

Although Algorithm 2 is not guaranteed to terminate at a global solution, our computational results in Section 7 indicate that its performance is quite satisfactory in the context of the inverse optimal value problem.

Remarks on the Proposed Strategy

Unless available a priori, computing \bar{c}^L in Algorithm 1 by solving the n linear programs (13) can be quite expensive. The following result establishes that this expensive step can be avoided.

Proposition 6.2. *If $\bar{c}^L \in C$ and Algorithm 2 is initialized with the solution (x^0, c^0) where c^0 is an optimal solution of problem (11) and $x^0 \in \arg \min_x \{c^{0T} x \mid Ax = b, x \geq 0\}$, then Algorithm 2 will terminate with a solution c' such that $Q(c') = Q(c^0)$ or $Q(c') < z^*$.*

Proof: Clearly Algorithm 2 will terminate with a solution $c' \in C$ satisfying $Q(c') \leq Q(c^0)$. Let us suppose that $z^* \leq Q(c') < Q(c^0)$. This implies that $c' \in \bar{C}$. Since $\bar{c}^L \in C$, by Proposition 5.1 we have that c^0 is a global optimal solution of problem (6), i.e., $Q(c^0) \leq Q(c')$. Hence we have a contradiction. □

By the above result, we can modify Algorithm 1, to avoid computing \bar{c}^L and checking $\bar{c}^L \in C$, as follows. If problem (11) is feasible, i.e., $\bar{C} \neq \emptyset$, we invoke Algorithm 2 for solving the bilinear program (16) with c^0 , the optimal solution of problem (11), as the initial solution. If indeed $\bar{c}^L \in C$, then we know that c^0 is a global solution of (2). If Algorithm 2 terminates with a solution c' such that $Q(c') = Q(c^0)$ then we have an alternate global optimal solution c' to problem (2). If Algorithm 2 terminates with a solution c' such that $Q(c') < z^*$ then we can recover a global optimal solution by the parametric program (17) as outlined in Algorithm 1.

Also note that it is not always necessary to execute Algorithm 2, the bilinear programming subroutine in Algorithm 1, to termination. If at any point in this

subroutine we obtain a solution (x, c) satisfying $c^T x \leq z^*$, we can terminate and return to Algorithm 1.

Finally, as mentioned before, if $\bar{c}^L \notin C$, Algorithm 1, with Algorithm 2 as the bilinear programming routine, is not guaranteed to provide a global solution of problem (2). However our computational results in the next section indicate quite satisfactory performance.

7. COMPUTATIONAL RESULTS

The proposed solution strategy was implemented in C++ using the CPLEX7.0 linear programming library routines. All computations were carried out on a Pentium II 450MHz processor PC with 256 MB RAM running Windows 2000. The proposed strategy was tested on several instances of the inverse optimal value problem wherein the underlying linear programs were either generated randomly or taken from the NETLIB [12] standard test set. Next, we describe the details of the computational experiments for each of these two cases.

Randomly generated LPs

Here we describe experiments with instances of the inverse optimal value problem for randomly generated linear programs of the form $\min\{c^T x \mid x \in X\}$. The feasible region of the LP is restricted to a polytope of the form $X = \{x \mid Ax \leq b, 0 \leq x \leq x^U\}$ to guarantee boundedness. We considered several different problem sizes in terms of the number of columns n and the number of rows m in A . In each case, we chose $x^U = 100e$, where $e \in \mathbb{R}^n$ is a vector of ones. The elements of A and b are uniformly generated in the interval $[-50, 50]$. Only feasible instances of the LPs are considered. Furthermore, we also ensure that $0 \notin X$ to avoid generating redundant instances. The set of cost coefficients C is also restricted to be a polytope of the form $C = \{c \mid Bc \leq d, c^L \leq c \leq c^U\}$. The number of rows and columns in B are the same as that in A . We chose $c^L = -100e$ and $c^U = 100e$, and the elements of B and d are uniformly generated in the interval $[-50, 50]$. Only feasible cost coefficient sets are considered.

First, we consider instances of the inverse optimal value problem satisfying $\bar{C} = \emptyset$. Such instances are generated by setting $z^* = \min\{c^{*T} x \mid x \in X\}$ where $c^* = c^U + \varepsilon e$, with ε being a small positive scalar. This guarantees that $Q(c) < z^*$ for all $c \in C$, and therefore $\bar{C} = \emptyset$. Recall that in this case, the algorithmic procedure is guaranteed to find a global optimal solution, and reduces to solving two linear programs – first problem (11) is solved to check $\bar{C} = \emptyset$, and then problem (12) produces an optimal solution to the inverse optimal value problem. Table 1 presents the computational times for various problem sizes. For each problem size, we report the minimum, average, and maximum CPU time over 20 feasible instances.

Next we consider problem instance where $\bar{C} \neq \emptyset$ but $\bar{C} \subset C$. Such an instance is generated by setting $z^* = \min\{c^{*T} x \mid x \in X\}$ where c^* is a vector in C . In this case, the algorithm first solves problem (11) to determine that $\bar{C} \neq \emptyset$, then it computes \bar{c}^L to check if $\bar{c}^L \in C$, and finally invokes the bilinear programming algorithm if $\bar{c}^L \notin C$. Using Proposition 6.2 we also consider the improved version of the algorithm where we avoid computing \bar{c}^L by initializing the bilinear algorithm with the solution of problem (11). Tables 2 and 3 present the computational results for various problem sizes. Here T_1 is the time required by the original algorithm, T_2 is the time to compute \bar{c}^L , and T_3 is time required by the improved algorithm. Once

again the minimum, average, and maximum time over 20 instances are reported. The CPU advantage in avoiding the \bar{c}^L computation is clear from these results. Although the bilinear algorithm is not guaranteed to converge to a global optimal solution, in our experiments the algorithm converged to the known global solution c^* for each generated instance. Thus, the proposed algorithm was able to solve to global optimality 100×100 instances of the inverse optimal value problem in less than 8 seconds.

Finally, we consider instances where $\bar{C} \neq \emptyset$ and $\bar{C} = C$. Such instances are generated by setting $z^* = \min\{c^{*T}x \mid x \in X\}$ where $c^* = c^L - \varepsilon e$. In this case global optimal solutions are not known. We compare the performance of the proposed algorithm to a straight forward mixed-integer programming (MIP) model for computing a global optimal solution to the inverse optimal value problem. The MIP formulation is described in the Appendix. Similar MIP reformulations are suggested in [5, 13]. Tables 4 and 5 presents the computational results. The first two columns of the tables indicate the problem size considered. We considered problem sizes of up to 28 columns and 16 rows, since the MIP approach required an excessive amount of computational time for larger instances. For each problem size, we generated 10 instances. Column 4 of each table presents the minimum, average, and maximum objective function optimality gap of the proposed strategy with respect to the MIP solution. Columns labeled T_1 , T_2 , and T_3 present the minimum, average and maximum time required by the original algorithm, to compute \bar{c}^L , and the improved algorithm, respectively. The column labeled T_m reports the minimum, average, and maximum time required to solve the MIP formulation using the CPLEX7.0 MIP solver. Finally, the last column indicates the average number of iterations required by the bilinear algorithm. The computational results indicate that the proposed algorithm is significantly faster than the MIP approach. Although the optimality gap can be as high as 29%, the proposed approach appears to perform quite satisfactorily returning solutions to within 12% of optimality on average.

Standard LPs

In this section, we consider instances of the inverse optimal value problem generated from five standard LP test problems from the NETLIB repository [12]. The specifications of these test problems are presented in Table 6. For each LP test problem, let c^* be its cost-coefficient data. We then generate an instance of the inverse optimal value problem such that c^* is its global optimal solution as follows. The set of cost coefficients C is restricted to be a polytope of the form $C = \{c \mid Bc \leq d, c^* - 1000e \leq c \leq c^* + 1000e\}$. The number of columns in B are the same as that in LP test problem. The number of rows are varied to generate problems of different size. The elements of B and d are uniformly generated in the interval $[-50, 50]$. To ensure that $c^* \in C$, if we find that the i th constraint satisfies $B_i c^* \geq d_i$, then we replace the constraint with $-B_i c \leq -d_i$. Only feasible instances of C are considered. We set z^* equal to the optimal value of the test LP.

First we attempt to solve these five problems using the MIP approach described in the Appendix. Table 7 presents the computational times for the case when the set C is defined by just 10 constraints. Clearly the MIP approach is impractical. Next, we test the proposed algorithm for the five problems with the number of rows defining C to vary from 10 to 100. Table 8 presents the computational results. For

each test problem, we present the minimum, average, and maximum computational time over 10 instances. The percent of problems for which the proposed strategy converged to the global optimal solution c^* is also presented. The proposed strategy was able to solve each of the problem instances to global optimality in less than 3 minutes.

TABLE 1. Computational time experiment result for the case $\bar{C} = \emptyset$

Rows	CPU s	Columns									
		10	20	30	40	50	60	70	80	90	100
10	Min	0.0	0.0	0.0	0.0	0.1	0.1	0.1	0.1	0.1	0.1
	Ave	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.2
	Max	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.2
20	Min	0.0	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.3	0.3
	Ave	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.3	0.4	0.4
	Max	0.5	0.1	0.1	0.1	0.2	0.2	0.3	0.4	0.5	0.6
30	Min	0.0	0.1	0.1	0.1	0.2	0.2	0.2	0.4	0.5	0.6
	Ave	0.1	0.1	0.1	0.2	0.2	0.3	0.4	0.6	0.6	0.7
	Max	0.1	0.2	0.3	0.2	0.3	0.4	0.6	0.7	0.9	0.9
40	Min	0.1	0.1	0.1	0.2	0.2	0.3	0.4	0.5	0.7	1.0
	Ave	0.1	0.1	0.2	0.2	0.3	0.4	0.6	0.7	0.9	1.2
	Max	0.4	0.2	0.3	0.3	0.3	0.5	0.7	0.9	1.1	1.5
50	Min	0.1	0.1	0.2	0.2	0.3	0.4	0.5	0.7	0.9	1.2
	Ave	0.1	0.1	0.2	0.2	0.3	0.5	0.6	0.8	1.2	1.5
	Max	0.2	0.2	0.3	0.3	0.5	0.5	0.7	1.0	1.4	1.7
60	Min	0.1	0.1	0.2	0.3	0.3	0.4	0.7	0.8	1.0	1.2
	Ave	0.1	0.2	0.2	0.3	0.4	0.5	0.8	1.0	1.3	1.6
	Max	0.3	0.6	0.5	0.4	0.4	0.7	0.9	1.1	1.7	2.1
70	Min	0.1	0.1	0.2	0.3	0.4	0.4	0.5	1.1	1.1	1.4
	Ave	0.1	0.2	0.2	0.3	0.4	0.5	0.9	1.2	1.4	1.7
	Max	0.5	0.3	0.4	0.5	0.7	0.9	1.8	1.3	1.6	2.0
80	Min	0.1	0.2	0.2	0.3	0.4	0.5	0.6	0.7	1.2	1.8
	Ave	0.1	0.2	0.3	0.4	0.5	0.6	0.9	1.2	1.5	2.1
	Max	0.2	0.5	1.0	0.8	1.0	0.8	1.1	1.4	2.1	2.3
90	Min	0.1	0.2	0.2	0.3	0.5	0.5	0.7	0.8	0.9	2.1
	Ave	0.1	0.2	0.3	0.4	0.5	0.7	0.9	1.3	1.5	2.4
	Max	0.5	0.3	0.4	0.7	0.7	1.0	1.4	2.5	1.7	2.8
100	Min	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.9	1.0	2.9
	Ave	0.1	0.2	0.3	0.4	0.6	0.8	1.0	1.3	1.8	2.9
	Max	0.3	0.3	0.5	0.7	1.1	1.1	1.2	1.4	2.3	2.9

TABLE 2. Computational experiment result for the case $\emptyset \subset \bar{C} \subset C$

Rows		Columns														
		10			20			30			40			50		
		T_1	T_2	T_3	T_1	T_2	T_3	T_1	T_2	T_3	T_1	T_2	T_3	T_1	T_2	T_3
10	Min	0.3	0.1	0.2	0.5	0.3	0.2	1.0	0.7	0.2	1.2	0.9	0.3	1.6	1.3	0.3
	Ave	0.4	0.2	0.2	0.7	0.4	0.3	1.4	1.1	0.3	1.7	1.3	0.3	2.4	2.0	0.4
	Max	0.6	0.4	0.4	1.1	0.8	0.4	2.8	2.5	0.6	2.2	1.8	0.7	3.6	3.3	0.5
20	Min	0.8	0.3	0.5	1.4	0.9	0.5	2.1	1.5	0.6	2.9	2.2	0.7	4.3	3.5	0.8
	Ave	0.9	0.4	0.6	1.8	1.2	0.6	2.8	2	0.7	3.7	2.9	0.8	5.2	4.4	0.8
	Max	1.2	0.6	0.7	2.2	1.6	0.9	3.6	2.7	1.0	4.9	4.0	1.0	6.6	5.8	1.0
30	Min	1.0	0.4	0.5	1.6	1.0	0.5	2.5	1.7	0.7	3.7	2.8	0.8	5.0	4.0	0.9
	Ave	1.1	0.5	0.6	2.1	1.4	0.7	3.1	2.3	0.8	4.4	3.5	0.9	6.0	4.9	1.1
	Max	1.5	0.9	0.8	2.7	1.7	1.0	4.2	3.4	1.0	5.1	4.2	1.7	6.8	5.7	1.7
40	Min	1.0	0.4	0.5	2.0	1.3	0.7	3.0	2.1	0.8	4.0	3.0	0.9	5.7	4.6	1.1
	Ave	1.1	0.5	0.6	2.3	1.6	0.7	3.9	3.1	0.9	5.5	4.5	1.0	7.0	5.8	1.2
	Max	1.5	0.8	0.8	2.7	2.0	0.9	4.7	3.9	1.2	7.3	6.3	1.3	9.4	8.2	1.6
50	Min	1.0	0.5	0.5	2.2	1.5	0.7	4.0	2.9	0.9	4.4	3.3	1.0	6.1	4.5	1.2
	Ave	1.2	0.6	0.6	2.5	1.8	0.8	4.6	3.6	1.0	6.9	5.7	1.2	9.3	7.9	1.5
	Max	1.5	0.9	0.8	3.1	2.0	1.1	5.6	4.6	1.2	10.0	8.9	1.6	14.0	12.0	2.0
60	Min	1.0	0.5	0.6	2.3	1.5	0.7	4.6	3.6	1.0	5.4	4.1	1.2	7.3	5.6	1.5
	Ave	1.3	0.6	0.7	2.7	1.9	0.8	5.3	4.1	1.1	9.0	7.6	1.3	12.0	10.4	1.9
	Max	1.7	1.0	0.8	3.3	2.5	0.9	6.4	5.2	1.8	11.0	9.3	1.6	16.0	14.0	2.5
70	Min	1.1	0.5	0.6	2.3	1.5	0.8	5.2	4.0	1.1	9.0	7.3	1.3	12.0	10.1	1.7
	Ave	1.3	0.6	0.7	2.8	1.9	1.0	5.8	4.5	1.2	10.0	8.4	1.7	16.0	14.2	1.9
	Max	1.7	1.1	1.1	3.5	2.3	1.5	6.8	5.4	1.5	12.0	9.5	2.5	19.0	17.3	2.5
80	Min	1.2	0.5	0.6	2.4	1.5	0.9	5.3	4.1	1.1	10.0	8.7	1.4	12.0	9.5	1.9
	Ave	1.4	0.6	0.7	3.0	2.1	1.0	6.3	5.0	1.4	12.0	9.8	1.9	18.0	15.9	2.3
	Max	1.8	0.8	1.1	3.5	2.6	1.1	7.1	5.9	1.8	13.0	11.0	2.7	20.0	17.4	3.1
90	Min	1.2	0.5	0.7	2.8	1.7	0.9	5.9	4.7	1.2	11.0	9.3	1.6	18.0	16.0	2.0
	Ave	1.5	0.7	0.8	3.2	2.2	1.0	6.7	5.3	1.4	12.0	11.0	1.8	21.0	18.5	2.3
	Max	2.0	1.1	1.0	4.6	3.1	1.6	8.3	6.4	1.9	14.0	12.0	2.2	23.0	21.2	3.3
100	Min	1.2	0.5	0.7	2.9	1.8	1.0	6.3	4.9	1.2	12.0	10.0	1.8	21.0	19.2	2.2
	Ave	1.4	0.7	0.8	3.4	2.3	1.1	7.2	5.6	1.6	14.0	12.0	2.1	23.0	20.5	2.6
	Max	1.9	1.1	1.0	4.1	3.0	1.4	8.1	6.6	2.1	16.0	14.0	2.8	26.0	23.0	3.5

TABLE 3. Computational experiment result for the case $\emptyset \subset \bar{C} \subset C(\text{con.})$

Rows		Columns														
		60			70			80			90			100		
		T_1	T_2	T_3	T_1	T_2	T_3	T_1	T_2	T_3	T_1	T_2	T_3	T_1	T_2	T_3
10	Min	2.7	2.4	0.4	2.8	2.4	0.4	4.6	4.1	0.5	5.6	5.0	0.5	5.1	4.4	0.5
	Ave	3.7	3.3	0.4	3.8	3.4	0.5	5.9	5.3	0.6	7.2	6.6	0.6	6.5	5.7	0.7
	Max	4.9	4.5	0.7	5.3	4.9	0.6	7.9	7.2	0.8	9.0	8.5	0.8	8.5	7.5	1.1
20	Min	5.2	4.4	0.8	6.5	5.3	0.9	9.0	7.9	1.0	10.4	8.9	1.1	8.2	7.2	0.9
	Ave	6.3	5.4	1.0	7.7	6.6	1.1	11.0	9.4	1.1	11.7	10.4	1.3	10.5	9.4	1.1
	Max	7.4	6.3	1.3	9.2	7.8	1.4	12.0	11.0	1.8	13.1	11.6	2.0	13.5	12.6	1.6
30	Min	6.7	5.6	1.0	8.6	7.4	1.2	11.0	9.8	1.3	12.8	11.4	1.4	12.6	11.1	1.2
	Ave	7.9	6.7	1.2	10.2	8.9	1.3	14.0	12.4	1.6	15.5	13.8	1.7	14.8	13.3	1.4
	Max	9.5	7.9	1.7	11.7	10.4	1.7	17.0	15.3	2.4	18.3	16.4	2.7	16.6	15.2	2.1
40	Min	7.9	6.3	1.2	10.8	9.1	1.4	13.0	11.1	1.7	16.2	14.4	1.7	16.5	14.8	1.6
	Ave	9.6	8.2	1.4	12.9	11.2	1.7	17.0	14.9	2.1	19.9	17.7	2.2	20.0	18.0	2.0
	Max	11.7	10.1	1.8	17.5	15.3	2.7	21.0	18.1	2.8	24.8	22.0	2.8	24.8	22.3	2.8
50	Min	8.3	6.5	1.4	11.0	9.4	1.6	15.0	12.7	1.9	21.0	18.7	2.3	21.9	18.8	2.1
	Ave	11.1	9.3	1.8	14.8	12.8	2.0	20.0	17.5	2.4	26.0	23.3	2.7	26.3	23.6	2.6
	Max	14.8	13.2	2.6	19.1	17.3	3.5	26.0	23.4	3.3	31.5	29.2	4.1	32.3	29.7	4.2
60	Min	10.1	8.1	1.7	12.3	10.4	1.9	18.0	15.3	2.4	21.8	18.9	2.6	24.6	21.5	2.5
	Ave	16.0	13.9	2.1	17.8	15.5	2.3	23.0	20.1	2.8	31.6	28.2	3.3	31.6	28.5	3.1
	Max	24.7	22.8	3.2	33.0	30.3	3.2	30.0	27.6	3.6	37.9	34.7	4.5	37.9	34.9	4.6
70	Min	10.5	8.6	1.8	12.6	10.2	2.1	18.0	14.9	2.6	25.8	22.5	3.0	21.2	18.6	2.6
	Ave	20.1	17.8	2.3	21.5	18.9	2.6	27.0	23.9	3.3	33.9	30.4	3.5	35.4	32.0	3.4
	Max	27.8	25.1	2.9	36.5	33.6	3.4	52.0	49.1	4.4	44.3	39.0	5.3	43.8	40.1	4.4
80	Min	13.0	10.4	2.2	16.8	14.2	2.4	18.0	15.0	2.9	27.7	23.3	3.1	26.5	22.6	3.5
	Ave	26.3	23.6	2.7	32.3	29.3	3.0	34.0	30.6	3.5	41.8	37.5	4.3	41.7	37.5	4.2
	Max	32.3	28.8	3.6	46.0	43.4	3.6	61.0	57.1	4.5	54.3	48.5	5.8	56.3	52.1	5.7
90	Min	13.4	10.8	2.4	18.1	15.3	2.8	25.0	21.3	3.3	28.4	23.5	3.4	29.5	25.7	3.8
	Ave	30.5	27.5	3.0	40.2	36.6	3.6	54.0	50.3	4.0	47.8	43.4	4.4	52.9	48.5	4.4
	Max	35.8	33.3	4.5	50.3	46.4	4.8	79.0	74.8	5.2	97.6	93.8	5.5	119.0	114.0	5.2
100	Min	33.1	29.6	2.8	26.9	23.3	3.4	26.0	22.2	4.0	27.1	22.0	3.7	27.8	23.8	3.8
	Ave	36.5	33.3	3.3	52.2	48.1	4.1	67.0	62.3	4.9	62.2	57.5	4.7	53.2	48.2	5.0
	Max	39.8	36.0	4.5	60.3	56.2	6.0	84.0	78.6	7.1	113.1	108.0	6.3	116.0	111.0	7.3

TABLE 4. Computational experiment result for the case $\bar{C} = C$

Columns	Rows		gap %	T_1	T_2	T_3	T_m	Iteration
4	4	Min	0.0	0.10	0.00	0.10	0.00	3
		Ave	0.0	0.20	0.00	0.10	0.00	
		Max	0.0	0.20	0.00	0.10	0.10	
8	4	Min	0.0	0.20	0.10	0.10	0.00	7
		Ave	3.0	0.20	0.10	0.10	0.10	
		Max	23.0	0.30	0.10	0.20	0.10	
12	4	Min	0.0	0.20	0.10	0.10	0.10	8
		Ave	8.0	0.30	0.20	0.10	0.20	
		Max	23.0	0.40	0.30	0.20	0.40	
12	8	Min	0.0	0.40	0.10	0.10	0.10	6
		Ave	5.0	0.40	0.20	0.30	0.10	
		Max	29.0	0.50	0.30	0.40	0.10	
16	4	Min	0.0	0.30	0.20	0.10	0.10	9
		Ave	0.0	0.40	0.20	0.20	1.90	
		Max	0.0	0.50	0.30	0.30	4.50	
16	8	Min	0.0	0.40	0.20	0.20	0.10	5
		Ave	1.0	0.50	0.30	0.20	0.30	
		Max	3.0	0.70	0.40	0.30	0.50	
20	4	Min	0.0	0.30	0.20	0.10	0.90	6
		Ave	0.0	0.30	0.20	0.10	11.30	
		Max	0.0	0.40	0.30	0.10	25.10	
20	8	Min	0.0	0.40	0.20	0.10	0.20	8
		Ave	12.0	0.40	0.20	0.20	1.90	
		Max	20.0	0.50	0.30	0.20	4.80	

TABLE 5. Computational experiment result for the case $\bar{C} = C$ (cont.)

Columns	Rows		gap %	T_1	T_2	T_3	T_m	Iterations
20	12	Min	0.0	0.40	0.20	0.20	0.20	5
		Ave	9.0	0.50	0.30	0.20	0.40	
		Max	25.0	0.60	0.30	0.30	0.60	
24	4	Min	0.0	0.30	0.30	0.10	45.40	7
		Ave	0.0	0.40	0.30	0.10	115.70	
		Max	0.0	0.50	0.30	0.20	167.10	
24	8	Min	0.0	0.30	0.20	0.10	0.90	10
		Ave	1.0	0.40	0.30	0.20	33.80	
		Max	5.0	0.60	0.30	0.30	124.40	
24	12	Min	0.0	0.40	0.30	0.20	0.30	6
		Ave	5.0	0.50	0.30	0.20	5.50	
		Max	24.0	0.80	0.40	0.50	17.20	
28	4	Min	0.0	0.40	0.30	0.10	38.30	7
		Ave	0.0	0.50	0.30	0.20	577.20	
		Max	0.0	0.60	0.40	0.30	1032.50	
28	8	Min	0.0	0.40	0.30	0.10	2.30	10
		Ave	0.0	0.50	0.30	0.20	237.70	
		Max	0.0	0.60	0.40	0.30	1055.30	
28	12	Min	0.0	0.40	0.30	0.10	0.90	9
		Ave	0.0	0.50	0.40	0.20	44.40	
		Max	0.0	0.70	0.40	0.30	169.60	
28	16	Min	0.0	0.50	0.30	0.20	0.60	5
		Ave	6.0	0.60	0.40	0.20	4.00	
		Max	19.0	0.80	0.50	0.40	5.70	

TABLE 6. Standard linear programming problems

Samples	Rows of X	Columns	Nonzero
Brandy	221	249	2150
Degen2	445	534	4449
Bnl1	644	1175	6129
25fv47	822	1571	11127
Ganges	1310	1681	7021

TABLE 7. The MIP approach

Samples	Rows of C	MIP time
Brandy	10	3569 secs
Degen2	10	74825 secs
Bnl1	10	> 24 hrs
25fv47	10	> 24 hrs
Ganges	10	> 24 hrs

TABLE 8. Standard test problems

Rows		Brandy		Degen2		Bnl1		25fv47		Ganges	
		Time	% [†]	Time	%	Time	%	Time	%	Time	%
10	Min	1.4		6.4		13.4		34.5		42.0	
	Ave	1.5	100	6.9	100	14.5	100	37.1	100	46.0	100
	Max	1.8		7.7		16.7		40.5		48.6	
20	Min	1.7		7.4		14.9		38.0		51.4	
	Ave	1.9	100	7.8	100	17.1	100	41.6	100	53.4	100
	Max	2.1		8.4		19.3		48.0		56.9	
30	Min	1.9		8.3		17.4		42.8		59.3	
	Ave	2.3	100	8.6	100	18.9	100	48.0	100	62.2	100
	Max	3.0		8.9		20.6		53.5		66.8	
40	Min	2.3		9.3		19.7		46.7		65.3	
	Ave	2.6	100	10.6	100	21.3	100	51.8	100	72.5	100
	Max	3.0		12.7		22.8		58.5		85.6	
50	Min	2.4		10.3		20.7		50.9		73.2	
	Ave	2.6	100	11.0	100	22.6	100	56.0	100	83.3	100
	Max	3.0		12.3		24.0		63.1		92.0	
60	Min	2.6		11.4		23.9		52.0		85.1	
	Ave	3.1	100	12.2	100	26.3	100	60.5	100	98.6	100
	Max	4.9		13.5		28.1		69.8		105.6	
70	Min	2.9		12.2		26.8		59.4		95.5	
	Ave	3.7	100	13.4	100	28.6	100	65.9	100	107.8	100
	Max	4.6		14.2		31.8		72.6		129.7	
80	Min	3.3		13.7		30.5		65.0		109.0	
	Ave	3.8	100	14.4	100	32.4	100	69.1	100	118.0	100
	Max	4.3		15.9		34.8		73.1		128.5	
90	Min	3.5		14.8		34.2		67.5		114.4	
	Ave	3.8	100	15.7	100	36.6	100	73.4	100	134.4	100
	Max	4.4		16.3		41.2		78.2		153.1	
100	Min	4.0		15.1		34.6		70.0		137.7	
	Ave	4.8	100	16.7	100	38.2	100	77.6	100	147.0	100
	Max	5.9		18.7		40.1		85.5		161.9	

† Percentage of samples that converged to global optimal solutions.

APPENDIX: A MIXED-INTEGER PROGRAMMING FORMULATION

It is easily verified that an instance of the inverse optimal value problem (2) can be restated as the following mixed-integer program

$$\begin{aligned}
 \min_{u^+, u^-, c, x, y} \quad & u^+ + u^-, \\
 \text{s.t.} \quad & u^+ - u^- = \pi^T b - z^*, \\
 & c \in C, \\
 & Ax = b, \\
 & 0 \leq x \leq My, \\
 & M(y - e) \leq \pi^T A - c \leq 0, \\
 & y \in \{0, 1\}^n,
 \end{aligned}$$

where M is a sufficiently large number and $e \in \mathbb{R}^n$ is a vector of ones. The auxiliary variables u^+ and u^- are used to model the absolute value function, and the binary variables y along with the big- M are used to model the complementary slackness condition between the primal and dual representations of $Q(c)$.

REFERENCES

- [1] R. K. Ahuja and J. B. Orlin. Inverse optimization. *Operations Research*, 49:771–783, 2001.
- [2] F. A. Al-Khayyal. Generalized bilinear programming. Part I: Models, applications and linear programming relaxation. *European Journal of Operational Research*, 60:306–314, 1992.
- [3] C. Audet, P. Hansen, B. Jaumard, and G. Savard. A symmetrical linear maxmin approach to disjoint bilinear programming. *Mathematical Programming*, 85:573–572, 1999.
- [4] K. P. Bennett and O. L. Mangasarian. Bilinear separation of two sets in n-space. *Computational Optimization and Applications*, 2:207–227, 1993.
- [5] O. Berman, D. I. Ingco, and A. Odoni. Improving the location of minimax facilities through network modification. *Networks*, 24:31–41, 1994.
- [6] D. Burton, W. R. Pulleyblank, and Ph. L. Toint. The inverse shortest paths problem with upper bounds on shortest path costs. In *Network Optimization*, Pardalos *et al.* (eds.), Springer Verlag, Lecture Notes in Economics and Mathematical Systems, 450:156–171, 1997.
- [7] D. Burton and Ph. L. Toint. On an instance of the inverse shortest paths problem. *Mathematical Programming*, 53:45–61, 1992.
- [8] D. Burton and Ph. L. Toint. On the use of an inverse shortest paths algorithm for recovering linearly correlated costs. *Mathematical Programming*, 63:1–22, 1994.
- [9] S. P. Fekete, W. Hochstättler, S. Kromberg, and C. Moll. The complexity of an inverse shortest paths problem. In *Contemporary Trends in Discrete Mathematics: From DIMACS and DIMATIA to the Future*, Graham *et al.* (eds.), DIMACS: Series in Discrete Mathematics and Theoretical Computer Science, 49:113–128, 1999.
- [10] C. Heuberger. Inverse optimization: A survey on problems, methods, and results. To appear in *Journal of Combinatorial Optimization*, 2002.
- [11] H. Konno. A cutting plane algorithm for solving bilinear programs. *Mathematical Programming*, 11:14–27, 1976.
- [12] NETLIB. LP Test Problems. www-fp.mcs.anl.gov/otc/Guide/TestProblems/LPtest/.
- [13] G. Paleologo and S. Takriti. Bandwidth trading: A new market looking for help from the OR community. *AIRO News*, VI(3):1–4, 2001.
- [14] H. D. Sherali and C. M. Shetty. A finitely convergent algorithm for bilinear programming problem using polar cuts and disjunctive face cuts. *Mathematical Programming*, 19:14–31, 1980.
- [15] T. V. Thieu. A note on the solution of bilinear problems by reduction to concave minimization. *Mathematical Programming*, 41:249–260, 1988.
- [16] H. Vaish and C. M. Shetty. A cutting plane algorithm for the bilinear programming problem. *Naval Research Logistics Quarterly*, 24:83–94, 1977.

- [17] D. J. White. A linear programming approach to solving bilinear programs. *Mathematical Programming*, 56:45–50, 1992.
- [18] J. Zhang and Z. Liu. Calculating some inverse linear programming problems. *Journal of Computational and Applied Mathematics*, 72:261–273, 1996.
- [19] J. Zhang and Z. Liu. A further study on inverse linear programming problems. *Journal of Computational and Applied Mathematics*, 106:345–359, 1999.
- [20] J. Zhang, Z. Liu, and Z. Ma. Some reverse location problems. *European Journal of Operational Research*, 124:77–88, 2000.

SCHOOL OF INDUSTRIAL & SYSTEMS ENGINEERING, GEORGIA INSTITUTE OF TECHNOLOGY, 765 FERST DRIVE, ATLANTA, GA 30332.