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 vectors, determine a cost vector such that the corresponding optimal objective value of the linear program is closest to the desired 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 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 vectors 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 $x^\star \in X$, find a cost vector $c^\star$ such that $x^\star$ is an optimal solution of P. Typically, $c^\star$ is required to satisfy some additional conditions. For example, given a preferred cost vector $\hat{c}$, the deviation $||c^\star - \hat{c}||_p$ 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 the ℓ_∞

Date: Submitted July 24, 2002. Revised June 9, 2003 and February 7, 2004.

* This research has been supported in part by the National Science Foundation under CAREER Award DMII-0133943. The authors thank two anonymous reviewers for valuable comments.

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

norm is also polynomially solvable. For a comprehensive survey of the literature on inverse optimization, the reader is referred to Heuberger [10].

In this paper, we consider the following generalization of the above standard inverse optimization problem: given the optimization problem P, a desired optimal objective value $z^\star$, and a set of feasible cost vectors C, determine a cost vector $c^\star \in C$ such that the corresponding optimal objective value of P is closest to $z^\star$. 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 time algorithm for the above inverse location problem on 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 cite an application in determining travel times in a road network from the observed travel times between some source-destination pairs. Fekete *et al.* [9] proved that this problem is NP-complete, and identified some polynomially solvable cases.

This paper is motivated in part by an application in telecommunication bandwidth pricing as proposed by Paleologo and Takriti [13]. Consider a bandwidth provider trying to price out city-to-city links on its bandwidth network. Suppose the observed traded price of a bandwidth link between two cities s and t in this network is z_{st}. Then the link prices should be such that the total price of the links on the cheapest path between s and t should be close to z_{st}. Otherwise an *arbitrage* opportunity opens up where a prudent trader can buy the cheaper of the two options – buying the links on the cheapest path or the traded link between city s and t – and sell the expensive option. Paleologo and Takriti [13] suggested a mixed-integer programming formulation for this bandwidth pricing problem. Note that this problem is an optimization version of the feasibility problem considered by Fekete *et al.* [9]. It should be pointed out that the bandwidth pricing problem in [13] is essentially an inverse *multi* optimal value problem – one wishes to find link prices such that the price of the cheapest path between city s and t, *i.e.*, the optimal value of the shortest $s-t$ path problem, is close to z_{st} for *all* such $s-t$ pairs. On the other, the problem considered in this paper is an inverse *single* optimal value problem where we are seeking an objective coefficient vector that matches a *single* optimal value function to a desired optimal value. Our study of inverse

single optimal value problems constitutes a first step towards the understanding of more general inverse multi optimal value problems.

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 an inverse optimal value problem where the underlying optimization problem is a linear program, and the set of cost vectors is restricted to a convex compact set. We show that the 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 vectors 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 the notation, and state some assumptions used throughout the paper.

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

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

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

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

where $f(c) := |Q(c) - z^\star|$ 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 $z^\star$. We shall denote such an instance by $\mathcal{P}(Q, C, z^\star)$.

Let us define the sets $C_{z^\star} := \{c \mid Q(c) \geq z^\star\}$ and $C_\infty := \{c \mid Q(c) > -\infty\}$. We shall frequently refer to the set $\overline{C} := C \cap C_{z^\star}$. Given a compact set C, let c^L be a point such that $c^L_j = \min\{c_j \mid c \in C\}$ for all $j = 1, \ldots, n$, and $\overline{c}^L$ be a point such that $\overline{c}^L_j = \min\{c_j \mid c \in \overline{C}\}$ for all $j = 1, \ldots, n$. Similarly, we let c^U be a point such that $c^U_j := \max\{c_j \mid c \in C\}$ for all $j = 1, \ldots, n$. 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 $\partial_L C := \Big\{c \in C \mid [c^L, c] \cap C = \{c\}\Big\}$. 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 $\overline{C}$. Finally, we define the notion of non-decreasing functions. Given two vectors a and b in $\mathbb{R}^n$, we write $a < b$ if $a_j \leq b_j$ for all $j = 1, \ldots, n$ and $a_{j'} < b_{j'}$ for some index j'. 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 $\{x \mid Ax = b,\ x \geq 0\}$ is non-empty.

(A2) The set of cost 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

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

and also $C_{z^\star} = \{c \mid \exists\ \pi \text{ s.t. } \pi^T A \leq c,\ \pi^T b \geq z^\star\}$ 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 over $\mathbb{R}^n$.* (ii) *$Q(\cdot)$ is piece-wise linear, concave and continuous over C_∞.* (iii) *The sets $C_{z^\star}$ 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 $B \in \mathbb{Z}^{m\times n}$, and an integer vector $d \in \mathbb{Z}^m$, the binary integer feasibility problem can be stated as follows

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

An instance of the binary integer feasibility problem is specified by the matrix-vector pair (B, d). We shall denote such an instance by $\mathcal{B}(B, d)$.

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

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

$$\begin{aligned} \hat{C} \ := \ & \Big\{(c_1, c_2, c_3)^T \in \mathbb{R}^{3n} \mid c_1 \in \mathbb{R}^n, c_2 \in \mathbb{R}^n, c_3 \in \mathbb{R}^n, \\ & Bc_1 \leq d,\ c_1 = c_2,\ c_3 = e, \\ & 0 \leq c_1 \leq e,\ 0 \leq c_2 \leq e \Big\}, \end{aligned}$$

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

$$\begin{aligned} \hat{Q}(c) := \quad & \min \quad c_1^T u - c_2^T v + c_3^T v \\ & \text{s.t.} \quad u + v \geq e,\ u \in \mathbb{R}^n_+,\ v \in \mathbb{R}^n_+, \end{aligned}$$

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

Suppose $\mathcal{B}(B,d)$ has an answer "yes," *i.e.*, there exists $\hat{x} \in \{0,1\}^n$ such that $B\hat{x} \leq d$. 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

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

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

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

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

where

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

Since $0 \leq \hat{c}_j \leq 1$, the optimal value of the above linear program will satisfy $\hat{Q}_j(\hat{c}) = \min\{\hat{c}_j, 1-\hat{c}_j\}$ for all j. Furthermore, $\hat{Q}(\overline{c}) = 0$ 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 $\mathcal{B}(B,d)$. □

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. □

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 $\overline{C}$ is empty or non-empty.

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

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

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

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

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

$$\begin{array}{ll} \max_{c,\pi} & b^T\pi, \\ \text{s.t.} & c \in C, \\ & \pi^T A - c \leq 0. \end{array} \tag{5}$$

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

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

$$\min\{Q(c) \mid c \in \overline{C}\}, \tag{6}$$

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

Proof: We first argue that there exists a solution $c^\star$ to the inverse optimal value problem (2) such that $c^\star \in \overline{C}$. Suppose the claim is not true. Then there exists $c^\star \in C \backslash \overline{C}$ such that $f(c^\star) \leq f(c)$ for all $c \in C$. Note that $c^\star \in C_\infty$, otherwise it is not an optimal solution. Since $c^\star \notin \overline{C}$, we have $Q(c^\star) < z^\star$. Now, consider a point $c' \in \overline{C}$. Note that $Q(c') \geq z^\star > Q(c^\star)$. Define $c^\lambda := \lambda c^\star + (1-\lambda)c'$ for any $\lambda \in (0,1)$. Since both $c^\star$ and c' are in the convex set $C \cap C_\infty$, so is c^λ for any $\lambda \in (0,1)$. Also, by Proposition 2.1(ii), $Q(\cdot)$ is concave and continuous over $[c^\star, c']$. By concavity of $Q(\cdot)$ we have $Q(c^\lambda) \geq \lambda Q(c^\star) + (1-\lambda)Q(c') > Q(c^\star)$ for any $\lambda \in (0,1)$. By continuity of $Q(\cdot)$ we can choose λ sufficiently close to 1 such that $Q(c^\lambda) \leq z^\star$. Thus $f(c^\lambda) = z^\star - Q(c^\lambda) < z^\star - Q(c^\star) = f(c^\star)$. Therefore $c^\star$ cannot be an optimal solution to problem (2). Thus, if $\overline{C} \neq \emptyset$ there exists an optimal solution $c^\star$ to (2) such that $c^\star \in \overline{C}$. Note that $Q(c) \geq z^\star$ for all $c \in \overline{C}$, thus problem (2) reduces to problem (6), and the claim follows. □

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 $\overline{C}$, it can be easily verified that problem (6) is equivalent to the following problem:

$$\begin{array}{ll} \min_{c,x,\pi} & c^T x, \\ \text{s.t.} & c \in C, \\ & \pi^T A - c \leq 0,\ \pi^T b \geq z^\star \\ & Ax = b,\ x \geq 0. \end{array} \tag{7}$$

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 $\overline{C} \neq \emptyset$, there exists a global solution to the inverse optimal value problem that is an extreme point on the "lower boundary" of $\overline{C}$. In Section 5, this characterization will suggest sufficient conditions under which the inverse optimal value problem is polynomially solvable.

The following lemma follows from the fact that $\overline{C}$ is a compact convex set.

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

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

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

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

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 $\overline{C} \neq \emptyset$. 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 $\overline{C}$. If we have that $\overline{c}^L \in \overline{C}$ then it is easily verified that $\partial_L \overline{C} \cap \Omega(\overline{C}) = \{\overline{c}^L\}$. Consequently, $\overline{c}^L$ is an optimal solution of (6), and hence of the inverse optimal value problem (2).

In this section we state more general conditions guaranteeing easy solvability of the inverse optimal value problem. In addition to assumptions (A1)-(A3), we shall also require the following assumption

(A4) $C \subseteq C_\infty$.

Assumption (A4) guarantees that the underlying linear program is bounded for all cost vectors in C. The assumption is trivially satisfied when the feasible region $\{x \mid Ax = b,\ x \geq 0\}$ of the underlying LP is bounded.

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

$$\min\{e^T c \mid c \in \overline{C}\}. \tag{8}$$

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

Proof: Consider first the case when $\overline{c}^L \in \overline{C}$. Then from Proposition 4.3, it follows that $\overline{c}^L$ is an optimal solution to problem (6) (since $\partial_L \overline{C} \cap \Omega(\overline{C}) = \{\overline{c}^L\}$) and, hence, is an optimal solution to the inverse optimal value problem (2). The claim then follows from noting that in this case $c^\star = \overline{c}^L$ is the unique optimal solution of (8).

Now consider the case that $\overline{c}^L \notin \overline{C}$. Suppose that the claim is not true. Then $z^\star < Q(c^\star)$. Since $\overline{c}^L \notin \overline{C}$, we have $-\infty < Q(\overline{c}^L) < z^\star < Q(c^\star)$ where the first

inequality is a consequence of Assumption (A4). Since $\overline{c}^L \in C$, we have that $[\overline{c}^L, c^\star] \in C \cap C_\infty$. By Proposition 2.1(ii) $Q(\cdot)$ is continuous over $[\overline{c}^L, c^\star]$, so there exists $c' \in (\overline{c}^L, c^\star)$ such that $Q(c') = z^\star$. Then $c' \in \overline{C}$ and, since clearly $c' < c^\star$, we have $e^T c' < e^T c^\star$. Therefore $c^\star$ cannot be an optimal solution to (8). □

Problem (8) above is equivalent to

$$\begin{array}{ll} \min_{c,\pi} & e^T c, \\ \text{s.t.} & c \in C, \\ & \pi^T A - c \leq 0,\ \pi^T b \geq z^\star, \end{array} \tag{9}$$

and is a convex program with a linear objective, and can be solved quite efficiently. In particular, when C is polyhedral, problem (9) is simply a linear program.

Theorem 5.1. *If $\overline{c}^L \in C$, and the convex programs (4) and (8) can be solved in polynomial time, then the inverse optimal value problem (2) can be solved in polynomial time.*

Proof: If (8) can be solved in polynomial time, then we can verify whether the convex set $\overline{C} = \emptyset$ in polynomial time. If $\overline{C} \neq \emptyset$, then by Proposition 5.1, an optimal solution of (8) is an optimal solution of (2). If $\overline{C} = \emptyset$, Proposition 4.1 implies that an optimal solution to (2) can be found in polynomial time by solving the convex program (4). □

Theorem 5.2. *If $c^L \in C$, and the convex programs (4) and (8) can be solved in polynomial time, then the inverse optimal value problem (2) can be solved in polynomial time.*

Proof: As discussed in the proof of Theorem 5.1, we can recognize and deal with the case $\overline{C} = \emptyset$ in polynomial time. Therefore, suppose $\overline{C} \neq \emptyset$. If $Q(c^L) \geq z^\star$, then it immediately follows that $c^L = \overline{c}^L$ and c^L is an optimal solution. Suppose $Q(c^L) < z^\star$. Consider any solution $c' \in \overline{C}$, for example a solution of (8). Then $-\infty < Q(c^L) < z^\star \leq Q(c')$, where the first inequality follows from Assumption (A4) and the third inequality follows from the definition of c'. From the continuity of $Q(\cdot)$, we can find (in polynomial time) an optimal solution $c^\star \in [c^L, c']$ such that $Q(c^\star) = z^\star$ by line search. □

The line search strategy mentioned in the above proof can be efficiently executed through parametric linear programming as outlined in the next section.

6. Solving the polyhedral case

The analysis in Section 4 suggests that we can solve the inverse optimal value problem by first checking whether $\overline{C} = \emptyset$, and then solving the corresponding convex problem (5) or the non-convex problem (7). Furthermore, in case $\overline{C} \neq \emptyset$, we can refine this scheme by verifying the condition $\overline{c}^L \in C$, and accordingly solving the convex program (8). 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, *i.e.*, in addition to assumptions (A1)-(A4), we shall assume henceforth that

(A5) The set C is polyhedral, *i.e.*, $C = \{c \mid Bc \leq d\}$.

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 (9) which, under assumption (A5), reduces to the linear program:

$$\begin{array}{ll} \min_{c,\pi} & e^T c, \\ \text{s.t.} & Bc \leq d, \\ & \pi^T A - c \leq 0,\ \pi^T b \geq z^\star. \end{array} \tag{10}$$

If the above LP is infeasible, we conclude $\overline{C} = \emptyset$. Otherwise, $\overline{C} \neq \emptyset$, and we denote an optimal solution of (10) 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 (A5), reduces to the linear program:

$$\begin{array}{ll} \max_{c,\pi} & b^T \pi, \\ \text{s.t.} & Bc \leq d, \\ & \pi^T A - c \leq 0. \end{array} \tag{11}$$

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

$$\begin{array}{lll} \overline{c}^L_j = & \min_{c,\pi} & e_j^T c, \\ & \text{s.t.} & Bc \leq d, \\ & & \pi^T A - c \leq 0,\ \pi^T b \geq z^\star \end{array} \tag{12}$$

for each $j = 1, \ldots, n$. In the above problem, 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 (10)) 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 (A5), reduces to the disjoint bilinear program:

$$\begin{array}{ll} \min_{c,x,\pi} & c^T x, \\ \text{s.t.} & Bc \leq d, \\ & \pi^T A - c \leq 0,\ \pi^T b \geq z^\star \\ & Ax = b,\ x \geq 0. \end{array} \tag{13}$$

Note that the above problem is stated in terms of the variables c, x, and π. Assumption (A4) allows for substantial simplification of (13). Recall that problem (13) is equivalent to $\min\{Q(c) \mid c \in \overline{C}\}$. Instead, let us consider the problem

$$\min\{Q(c) \mid c \in C\}. \tag{14}$$

Assumption (A4) guarantees that $Q(c)$ is continuous over C, hence problem (14) is well-defined. We can now re-state this problem as

$$\begin{array}{ll} \min_{c,x} & c^T x, \\ \text{s.t.} & Bc \leq d, \\ & Ax = b,\ x \geq 0. \end{array} \tag{15}$$

Problem (15) avoids inclusion of the π variables in the bilinear formulation and is significantly easier to solve than (13). Note, however, that a solution of (15) is no longer guaranteed to satisfy $Q(c) \geq z^\star$, and is, therefore, not necessarily an optimal solution of the inverse optimal value problem. Let (c', x') be a global optimal solution of problem (15). If $c'^T x' \geq z^\star$, then clearly $c' \in \overline{C}$ and is, therefore, a global

optimal solution of the inverse optimal value problem. Otherwise if $c'^T x' < z^\star$, then we have $Q(c') < z^\star \leq Q(c^0)$, where c^0 is an optimal solution of problem (10). By the continuity of $Q(c)$ over C we know that there exists $c^\star \in [c', c^0]$ such that $Q(c^\star) = z^\star$, hence $c^\star$ is an optimal solution of the inverse optimal value problem. Such a $c^\star$ 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

$$F(\lambda) = \min\{(c^0 + \lambda\Delta)^T x \mid Ax = b,\ x \geq 0\}, \tag{16}$$

with $\Delta = c' - c^0$, for $\lambda \in [0, 1]$ to find $\lambda^\star$ such that $F(\lambda^\star) = z^\star$. Then $c^\star = c^0 + \lambda^\star \Delta$ 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 (10).
if problem (10) is infeasible, i.e., C̄ = ∅ then
  solve the linear program (11), and let c* be its optimal solution.
else {problem (10) is feasible, i.e., C̄ ≠ ∅}
  let c^0 be an optimal solution of (10).
  compute c̄^L by solving the linear programs (12) for j = 1, ..., n.
  if c̄^L ∈ C then
    set c* ← c^0.
  else {c̄^L ∉ C}
    solve the bilinear program (15) and let (c', x') be an optimal solution.
    if c'^T x' ≥ z* then
      set c* ← c'.
    else {c'^T x' < z*}
      solve the parametric linear program (16) to find c* ∈ [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 (15). 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 (15)

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
  $x^{i+1} \in \arg \text{vertex partial } \min_x\{{c^i}^T x \mid Ax = b,\ x \geq 0\}$,
  $c^{i+1} \in \arg \text{vertex partial } \min_c\{{x^{i+1}}^T c \mid Bc \leq d\}$,
  and ${c^{i+1}}^T x^{i+1} < {c^i}^T x^i$.
  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 (15) or a solution (x^{i+1}, c^i) satisfying the necessary optimality condition: ${c^i}^T(x - x^{i+1}) + {x^{i+1}}^T(c - c^i) \geq 0$ 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 $\overline{c}^L$ in Algorithm 1 by solving the n linear programs (12) can be quite expensive. We can defer (and sometimes may be able to avoid) this computation by initializing the bilinear programming algorithm (Algorithm 2) appropriately.

Proposition 6.2. *If $\overline{c}^L \in C$ and Algorithm 2 is initialized with the solution (x^0, c^0) where c^0 is an optimal solution of problem (10) and $x^0 \in \text{argmin}_x\{{c^0}^T x \mid Ax = b,\ x \geq 0\}$, then Algorithm 2 will terminate with a solution c' such that $Q(c') = Q(c^0)$ or $Q(c') < z^\star$.*

Proof: Clearly Algorithm 2 will terminate with a solution $c' \in C$ satisfying $Q(c') \leq Q(c^0)$. Let us suppose that $z^\star \leq Q(c') < Q(c^0)$. This implies that $c' \in \overline{C}$. Since $\overline{c}^L \in C$, 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. □

The above result suggests that if Algorithm 2 is initialized with the solution (x^0, c^0), then we need to check the condition $\overline{c}^L \in C$ *only if* $Q(c') > z^\star$ *and* $Q(c') = Q(c^0)$. If Algorithm 2 terminates at a point c' such that $Q(c^0) > Q(c') > z^\star$, then Proposition 6.2 implies $\overline{c}^L \notin C$, and consequently, we do not need to calculate $\overline{c}^L$. In this case, we cannot guarantee a global optimal solution and conclude that c' is an approximate optimal solution with an optimality gap of no more than $(Q(c') - z^\star)$. If, on the other hand, $Q(c^0) = Q(c') > z^\star$, then we calculate $\overline{c}^L$ to see if $\overline{c}^L \in C$. If this condition is satisfied then we are guaranteed that c' is a global optimal solution. If $\overline{c}^L \notin C$, then we conclude that c' is an approximate optimal solution

with an optimality gap of no more than $(Q(c') - \max\{z^\star, Q(\overline{c}^L)\})$. This bound on the optimality gap follows from the fact that if $c^\star$ is an optimal solution, then we know that $Q(c^\star) \geq z^\star$ and $Q(c^\star) \geq Q(\overline{c}^L)$ (since $c^\star \in \overline{C}$). Following the preceding discussion, we can modify Algorithm 1 to Algorithm 3.

Algorithm 3 Modified Algorithm

solve the linear program (10).
if problem (10) is infeasible, *i.e.*, $\overline{C} = \emptyset$ **then**
  solve the linear program (11), and let $c^\star$ be its optimal solution ($c^\star$ is a global optimal solution to (2)).
else {problem (10) is feasible, *i.e.*, $\overline{C} \neq \emptyset$}
  let c^0 be an optimal solution of (10) and $x^0 \in \operatorname{argmin}_x\{{c^0}^T x \mid Ax = b,\ x \geq 0\}$.
  solve the bilinear program (15) using Algorithm 2 with (x^0, c^0) as the initial solution. Let $(c',\ x')$ be the solution returned by Algorithm 2.
  if ${c'}^T x' = z^\star$ **then**
    set $c^\star \leftarrow c'$ ($c^\star$ is a global optimal solution of (2)).
  else if ${c'}^T x' > z^\star$ **then**
    if ${c'}^T x' = {c^0}^T x^0$ **then**
      compute $\overline{c}^L$ by solving the linear programs (12) for $j = 1, \ldots, n$.
      if $\overline{c}^L \in C$ **then**
        set $c^\star \leftarrow c'$ ($c^\star$ is a global optimal solution of (2)).
      else $\{\overline{c}^L \notin C\}$
        set $c^\star \leftarrow c'$ ($c^\star$ is an approximate solution of (2) with an optimality gap $\leq ({c'}^T x' - \max\{z^\star, Q(\overline{c}^L)\})$.
      end if
    else $\{{c'}^T x' < {c^0}^T x^0\}$
      set $c^\star \leftarrow c'$ ($c^\star$ is an approximate solution of (2) with an optimality gap $\leq ({c'}^T x' - z^\star)$.
    end if
  else if ${c'}^T x' < z^\star$ **then**
    solve the parametric linear program (16) to find $c^\star \in [c', c^0]$ such that $Q(c^\star) = z^\star$ ($c^\star$ is a global optimal solution of (2)).
  end if
end if
return $c^\star$.

Finally, note that it is not always necessary to execute Algorithm 2, the bilinear programming subroutine, to termination. If at any point in this subroutine we obtain a solution (x', c') satisfying ${c'}^T x' \leq z^\star$, we can terminate and return.

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 consider several different problem sizes in terms of the number of columns n and the number of rows m in A. In each case, we choose $x^U = 100e$, 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 vectors C is also restricted to be a polytope of the form $C = \{c \mid Bc \leq d,\ l \leq c \leq u\}$. The number of rows and columns in B are the same as that in A. We choose $l = -100e$ and $u = 100e$, and the elements of B and d are uniformly generated in the interval $[-50, 50]$. Only feasible sets of cost vectors are considered.

First, we consider instances of the inverse optimal value problem satisfying $\overline{C} = \emptyset$. Such instances are generated by setting $z^\star = \min\{{c^\star}^T x \mid x \in X\}$ where $c^\star = u + \varepsilon e$, with ε being a small positive scalar. This guarantees that $Q(c) < z^\star$ for all $c \in C$, and therefore $\overline{C} = \emptyset$. 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 (10) is solved to check $\overline{C} = \emptyset$, and then problem (11) 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 instances where $\overline{C} \neq \emptyset$ but $\overline{C} \subset C$. Such instances are generated by setting $z^\star = \min\{{c^\star}^T x \mid x \in X\}$ where $c^\star$ is a vector in C. Thus $c^\star$ is a global optimal solution to the generated instance. For these instances, Algorithm 1 first solves problem (10) to resolve that $\overline{C} \neq \emptyset$, then it computes $\overline{c}^L$ to check if $\overline{c}^L \in C$, and finally invokes the bilinear programming algorithm if $\overline{c}^L \notin C$. We also consider Algorithm 3 where we defer computing $\overline{c}^L$ by initializing the bilinear algorithm with the solution of problem (10). Tables 2 and 3 present the computational results for various problem sizes. Here T_1 is the time required by Algorithm 1, T_2 is the time to compute $\overline{c}^L$, and T_3 is the time required by Algorithm 3. Once again the minimum, average, and maximum time over 20 instances is reported. Although the bilinear algorithm is not guaranteed to converge to a global optimal solution, in our experiments both algorithms converges to the known global solution $c^\star$ for each generated instance. Furthermore, Algorithm 3 never requires to check $\overline{c}^L \in C$ thereby avoiding a computationally expensive step. The CPU advantage in avoiding the $\overline{c}^L$ computation is clear from the tabulated results. Algorithm 3 is able to solve instances of the inverse optimal value problem of size 100×100 in less than 8 seconds to global optimality.

Finally, we consider instances where $\overline{C} \neq \emptyset$ and $\overline{C} = C$. Such instances are generated by setting $z^\star = \min\{{c^\star}^T x \mid x \in X\}$ where $c^\star = l - \varepsilon e$. In this case the global optimal solution to the generated instance is not known *a priori*. We compare the performance of the proposed algorithms to a straightforward 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 present the computational results. The first two columns of the tables indicate the problem

sizes considered. We consider problem sizes of up to 28 columns and 16 rows, since the MIP approach requires an excessive amount of computational time for larger instances. For each problem size, we generate 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 labelled T_1, T_2, and T_3 present the minimum, average and maximum time required by Algorithm 1, to compute $\overline{c}^L$, and Algorithm 3, respectively. In Algorithm 3 we avoid computing $\overline{c}^L$ altogether, and return c' as an approximate solution. The column labelled 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 algorithms are significantly faster than the MIP approach. Although the optimality gap can be as high as 29%, the proposed approaches appear 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 $c^\star$ be its cost vector. We then generate an instance of the inverse optimal value problem with $c^\star$ as its global optimal solution as follows. The set of cost vectors C is restricted to be a polytope of the form $C = \{c \mid Bc \leq d,\ c^\star - 1000e \leq c \leq c^\star + 1000e\}$. 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 sizes. The elements of B and d are uniformly generated in the interval $[-50, 50]$. To ensure that $c^\star \in C$, if we find that the ith constraint satisfies $B_i c^\star \geq d_i$, then we replace the constraint with $-B_i c \leq -d_i$. Only feasible instances of C are considered. We set $z^\star$ 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 Algorithm 3 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 $c^\star$ is also presented. The proposed strategy is able to solve each of the problem instances to global optimality in less than 3 minutes.

TABLE 1. Computational results for the case $\bar{C} = \emptyset$

		Columns									
Rows	CPU s	**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	0.2	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 results for the case $\emptyset \subset \bar{C} \subset C$

		Columns														
		10			**20**			**30**			**40**			**50**		
Rows		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
	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**
10	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**
	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**
20	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**
	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**
30	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**
	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**
40	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**
	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**
50	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**
	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**
60	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**
	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**
70	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**
	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**
80	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**
	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**
90	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**
	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**
100	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 results for the case $\emptyset \subset \bar{C} \subset C$ (contd.)

		Columns														
		60			**70**			**80**			**90**			**100**		
Rows		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
	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**
10	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**
	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**
20	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**
	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**
30	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**
	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**
40	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**
	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**
50	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**
	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**
60	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**
	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**
70	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**
	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**
80	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**
	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**
90	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**
	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**
100	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 results for the case $\bar{C} = C$

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

TABLE 5. Computational results for the case $\bar{C} = C$ (contd.)

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

TABLE 6. Standard linear programming problems

Samples	Rows of X	Columns	Nonzeros
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

		Brandy		Degen2		Bnl1		25fv47		Ganges	
Rows		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

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

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, Z. Ganz, and J. M. Wagner. A stochastic optimization model for planning capacity expansion in a service industry under uncertian demand. *Naval Research Logistics*, 41:545–564, 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 combinatorial optimization: A survey on problems, methods, and results. To appear in *Journal of Combinatorial Optimization*, 2003.

[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.