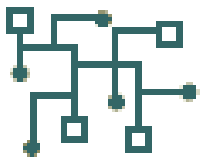
Inverse Integer Linear Programs: Complexity and Computation

Aykut Bulut[1] Ted Ralphs[1]

[1]COR@L Lab, Department of Industrial and Systems Engineering, Lehigh University

INFORMS Computing Society Conference, 6 January, 2013



ISE

Industrial and Systems Engineering

COR@L

COMPUTATIONAL OPTIMIZATION RESEARCH AT LEHIGH

Outline

Outline

What is an Inverse Problem?

We consider a mixed integer linear program (MILP)

$$z_{IP} = \min_{x \in \mathcal{P}} c^\top x, \tag{MILP}$$

where $c \in \mathbb{R}^n$ and

$$\mathcal{P} = \left\{ x \in \mathbb{R}^n_+ \mid\mid Ax = b \right\} \cap (\mathbb{Z}^r \times \mathbb{R}^{n-r}).$$

- Roughly speaking, a *parametric MILP* is the problem of finding the sequence of solutions that arise as we vary the input sytematically.
- The associated *inverse problem* reverses this process.
  - We are given a solution from the sequence and we want to determine what input would yield it.
  - Typically, we start with a target and try to find the closest input (by some norm) that yields the desirted output.
- Note that these concepts apply to optimization problems in general, but we only consider MILPs here.

Our Setting

- In this talk, we consider problems in which the objective function is parameterized.
- We refer to the original problem (MILP) as the *forward problem* and consider the original objective vector c to be the *target objective*.
- We denote the parametric version of (MILP) by

$$z_{IP}(d) = \min_{x \in \mathcal{P}} d^\top x$$

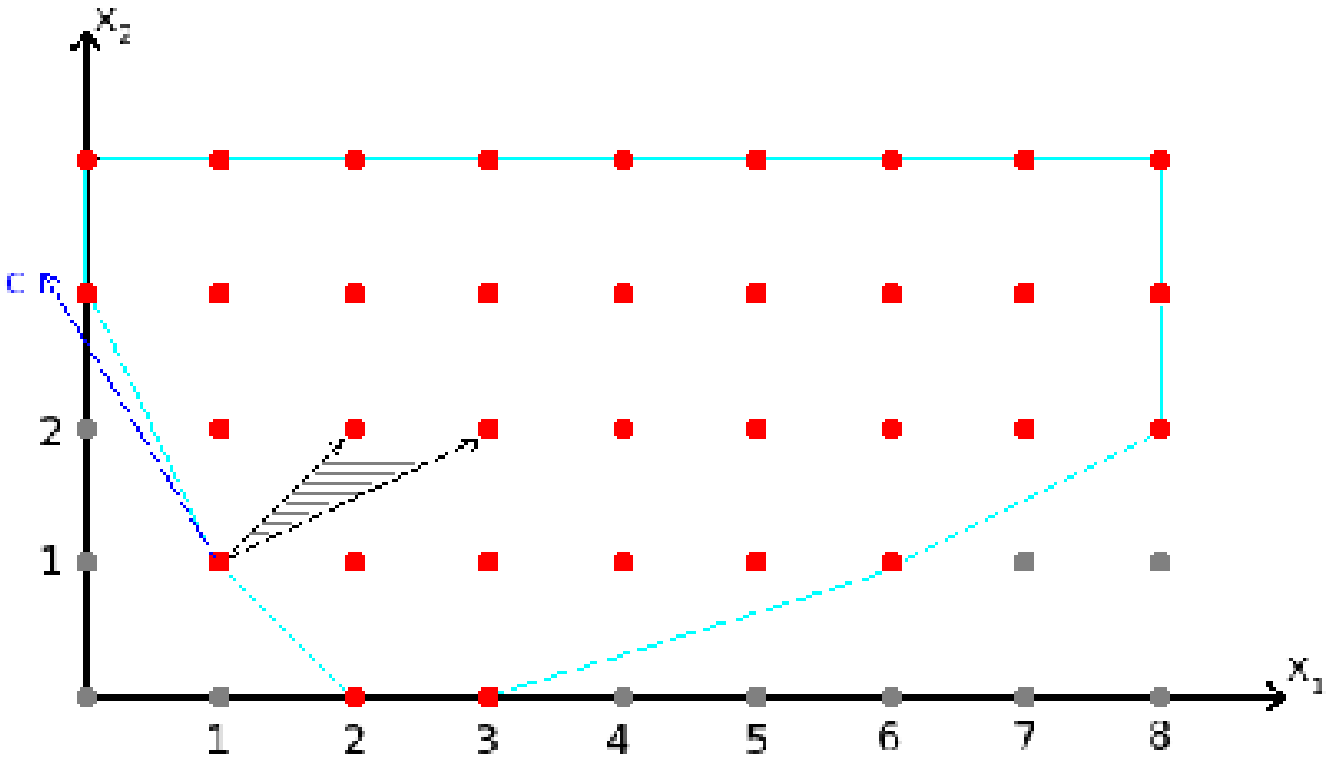
- The function z_{IP} here is the *value function* of (MILP).
- The associated *inverse problem* is further specificed by the *target solution* $x^0 \in \mathcal{P}$.
- The inverse problem is then to find

$$\operatorname{argmin} \left\{ \|c - d\| \quad \middle| \quad z_{IP}(d) = d^\top x^0 \right\}$$

A Small Example

Define forward problem feasible set $\mathcal{P}$ as follows and let $c = (-3, 5)$, $x^0 = (1, 1)$.

Figure : $\mathcal{P}$, its convex hull and feasible d cone



Applications

- Inverse shortest path problems, used in predicting the movement of earthquakes, arising in geophysical sciences [1].
- Inverse programs naturally arise in situations when we want to infer a person's utility function from the person's actions.
- In numerical analysis, errors for solving the system $Ax = b$ are measured in terms of inverse optimization.
- Any kind of system that we can observe the solutions but can not measure all the parameters that results in these solutions.

Formulating the Inverse Problem

For a given $c \in \mathbb{R}^n$, $x^0 \in \mathcal{P}$, the inverse problem can be stated as the following mathematical program.

$$\begin{aligned} \min \quad & \|c - d\| \\ \text{s.t.} \quad & d^\top x^0 \leq d^\top x \quad \forall x \in \mathcal{P}. \end{aligned}$$

- We assume here that $\mathcal{P}$ is bounded.
- Model can be linearized for l_1 and l_∞ norms.
- The convex hull of $\mathcal{P}$ is a polytope.
- Let $\mathcal{E}$ be the set of extreme points of convex hull of $\mathcal{P}$.
- The constraints set can be replaced by the finite set involving members of $\mathcal{E}$.

Inverse IP with l_1 norm

$$
\begin{aligned}
z^1_{IP} = \min \quad & \sum_{i=1}^{n} \theta_i & \\
\text{s.t.} \quad & c_i - d_i \leq \theta_i & \forall i \in \{1, 2, \ldots, n\} \\
& d_i - c_i \leq \theta_i & \forall i \in \{1, 2, \ldots, n\} \\
& d^\top x^0 \leq d^\top x & \forall x \in \mathcal{E}.
\end{aligned}
$$

Inverse IP with l_∞ norm

$$
\begin{aligned}
z_{IP}^{\infty} = \min \quad & y & \\
\text{s.t.} \quad & c_i - d_i \leq y & \forall i \in \{1, 2, \ldots, n\} \\
& d_i - c_i \leq y & \forall i \in \{1, 2, \ldots, n\} \\
& d^\top x^0 \leq d^\top x & \forall x \in \mathcal{E}.
\end{aligned}
$$

- For the remainder of the presentation, we deal with the case of l_∞ norm.
- Let $\mathcal{S}$ represent the feasible set of the inverse IP, defined as

$$\mathcal{S} = \left\{(y, d) \in \mathbb{R} \times \mathbb{R}^n | y \geq \|c - d\|_\infty, d^\top (x^0 - x) \leq 0 \quad \forall x \in \mathcal{E}\right\}.$$

- Note that $\mathcal{S}$ is a polyhedron.

Outline

Polynomially Solvable Forward Problems

- Ahuja and Orlin [1] determined the complexity of the inverse problem when the forward problem is polynomially solvable.
- The analysis can be easily extended to accomodate forward problems in other classes.

Theorem

(Ahuja and Orlin [1]) If a forward problem is polynomially solvable for each linear cost function, then corresponding inverse problems under l_1 and l_∞ are polynomially solvable.

Forward Problems

Define the following problems related to IP and inverse IP.

Definition

IP decision problem: Given $\gamma \in \mathbb{Q}$ decide whether $d^\top x \leq \gamma$ holds for some $x \in \mathcal{P}$.

Definition

IP optimization problem: Find solution vector x^* such that $x^* \in \text{argmin}_{x \in \mathcal{P}} d^\top x$ or decide the problem is unbounded or decide the problem is infeasible.

Inverse Problems

Definition

Inverse IP decision problem: Given $\gamma \in \mathbb{Q}$ decide whether $y \leq \gamma$ holds for some $(y, d) \in \mathcal{S}$.

Definition

Inverse IP optimization problem: Find solution vector (y^*, d^*), such that $(y^*, d^*) \in \text{argmin}_{(y,d)\in\mathcal{S}} y$.

Definition

Inverse IP separation problem: Given a vector $(\overline{y}, \overline{d}) \in \mathbb{Q} \times \mathbb{Q}^n$, decide whether $(\overline{y}, \overline{d}) \in \mathcal{S}$, and if not, find a hyperplane that separates $(\overline{y}, \overline{d})$ from $\mathcal{S}$, i.e., find $\pi \in \mathbb{Q}^{n+1}$ such that $\pi^\top \begin{bmatrix} \overline{y} \\ \overline{d} \end{bmatrix} < \min \left\{ \pi^\top \begin{bmatrix} y \\ d \end{bmatrix} | (y, d) \in \mathcal{S} \right\}$.

Equivalence of Optimization and Separation (GLS)

The following theorem by Grötschel et al. indicates the relationship between separation and optimization problems.

Theorem

(Grötschel et al. [2]) Given an oracle for the separation problem, the optimization problem over a given polyhedron with linear objective can be solved in time polynomial in φ, n, and the encoding length of objective coefficient vector, where facet complexity of polyhedron is at most φ.

Facet/Vertex Complexity

Definition

Facet complexity: A polyhedron $\mathcal{Q}$ has *facet-complexity at most* φ if there exists a system of inequalities that has solution set $\mathcal{P}$ and such that encoding length of each inequality of the system is at most φ.

Definition

Vertex complexity: A polyhedron $\mathcal{Q}$ has *vertex-complexity at most* ν if there exist finite sets V and E of rational vectors such that $\mathcal{S} = conv(V) + cone(E)$ and such that each of the vectors in V and E has encoding length at most ν. In case $\mathcal{S} = \emptyset$, $\nu \geq n$.

- If $\mathcal{S}$ has facet-complexity at most φ, then $\mathcal{S}$ has vertex-complexity at most $4n^2\varphi$
- If $\mathcal{S}$ has vertex-complexity at most ν, then $\mathcal{S}$ has facet-complexity at most $3n^2\nu$

Facet/Vertex Complexity

Recall that

$$\mathcal{S} = \left\{(y, d) \in \mathbb{R} \times \mathbb{R}^n | y \geq \|c - d\|_\infty, d^\top (x^0 - x) \leq 0 \quad \forall x \in \mathcal{E}\right\}.$$

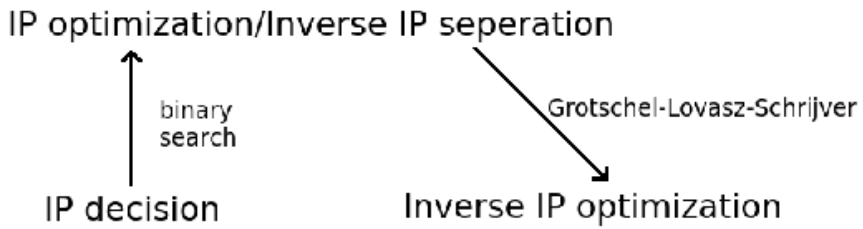
- The facet complexity of $\mathcal{S}$ is the encoding length of the inequalities $d^T(x^0 - x) \leq 0 \quad \forall x \in \mathcal{E}$.
- This is the facet complexity of the convex hull of $\mathcal{P}$.
- It is closely related to the vertex complexity of convex hull of $\mathcal{P}$ (the encoding length of members of $\mathcal{E}$).
- For combinatorial problems, this encoding length would generally be n and would be modest in most practical cases.

Complexity of IP optimization/decision problems

Recall that

$$\mathcal{S} = \left\{(y, d) \in \mathbb{R} \times \mathbb{R}^n | y \geq \|c - d\|_\infty, d^\top (x^0 - x) \leq 0 \quad \forall x \in \mathcal{E}\right\}.$$

- It is easy to see that the separation problem for $\mathcal{S}$ is an instance of the forward problem.
- Thus, by GLS, we see that the decision version of the forward problem and the inverse problem are polynomially equivalent.
- Similarly, the decision versions of both the forward and inverse problems are polynomially equivalent.
- But are they in the same complexity class?



The Polynomial Hierarchy

The polynomial hierarchy is a scheme for classifying multi-level and multi-stage decision problems. We have

$$\Delta_0^P := \Sigma_0^P := \Pi_0^P := P,$$

where P is the set of decision problems that can be solved in polynomial time. Higher levels are defined recursively as:

$$\begin{aligned} \Delta_{k+1}^P &:= P^{\Sigma_k^P}, \\ \Sigma_{k+1}^P &:= NP^{\Sigma_k^P}, and \\ \Pi_{k+1}^P &:= coNP^{\Sigma_k^P}. \end{aligned}$$

PH is the union of all levels of the hierarchy.

Δ_2^P and NP

- Where does inverse integer programming fall in the hierarchy?
- The decision version of inverse integer programming is an example of a problem in Δ_2^P.
- Such problem are polynomially equivalent to problems in NP, but are not themselves in NP.
- Intuitively, problems in NP are existentially quantified, whereas problems in Δ_2^P are universally quantified.
  - The decision version of the inverse problem requires proving a lower bound on the optimal value of the forward problem (*all* solutions have cost *greater* than or equal to a given target.
  - The decision version of the forward problem requires proving an redupper bound (at least one solution has cost less than or equal to a given target).
  - Optimization requires providing an upper and a lower bound.
- Higher levels of the hierarchy alternate between embedded universal and existentially quantified problems.

Complexity of Inverse Integer Programming

Theorem

Inverse IP optimization problem under l_∞/l_1 norm is solvable in time polynomial in φ, $n+1/2n$, and encoding length of $(1,0,\ldots,0)/(1,\ldots,1,0,\ldots,0)$, given an oracle for the IP decision problem.

Theorem

The inverse IP decision problem is in Δ_2^{P}.

Conjecture

The inverse IP decision problem is complete for Δ_2^{P}.

Outline

Proposed Algorithm

First, we define two parametric problems named P_k and $InvP_k$ as follows

$$\min_{x \in \mathcal{P}} d^{kT} x \qquad (P_k)$$

$$\begin{aligned} \min \quad & y & & & (InvP_k) \\ \text{s.t.} \quad & c_i - d_i \leq y & & \forall i \in \{1, 2, \ldots, n\} \\ & d_i - c_i \leq y & & \forall i \in \{1, 2, \ldots, n\} \\ & d^\top x^0 \leq d^\top x & & \forall \mathcal{E}^{k-1} \end{aligned}$$

where $\mathcal{E}^{k-1}$ is the set of extreme points found so far.

Proposed Algorithm

Algorithm 1

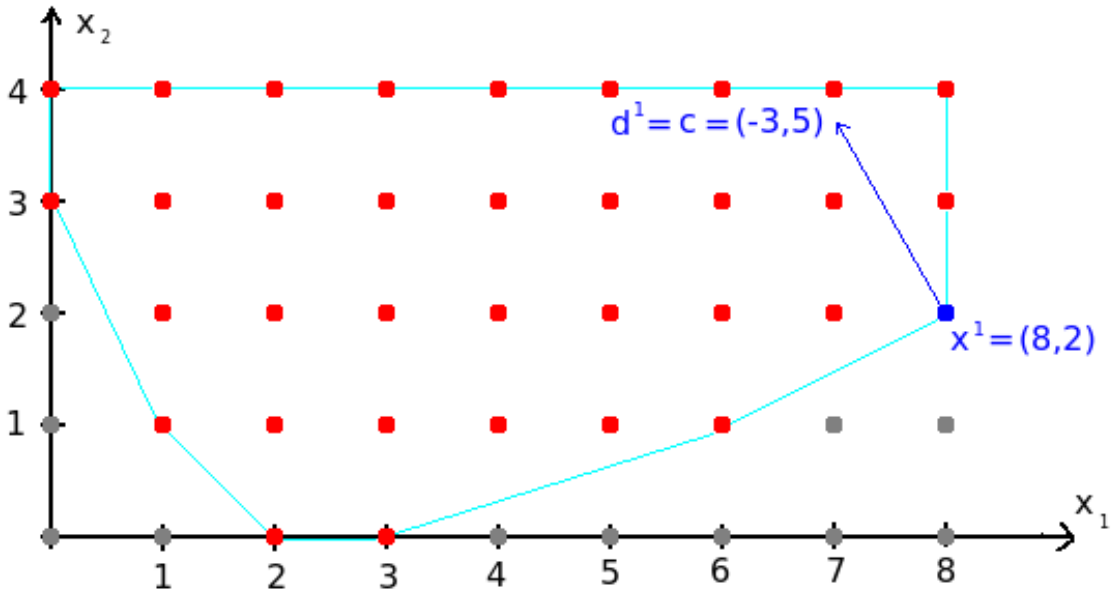
$k \leftarrow 1$
$d^k \leftarrow c$
Solve P_k, $x^k \leftarrow x^*$
while $d^{kT}(x^0 - x^k) > 0$ **do**
    $k \leftarrow k + 1$
    Solve $InvP_k$, $d^k \leftarrow d^*$
    Solve P_k, $x^k \leftarrow x^*$
end while

- x^k is an optimal solution to P_k, whereas d^k is an optimal solution to $invP_k$.
- The algorithm stops when a generated cut is not violated by the current solution.

A Small Example: Iteration 1

Let $\mathcal{P}$ and convex hull of $\mathcal{P}$ given as in Figure. Let $c = (-3, 5)$ and $x^0 = (1, 1)$. We know $d^1 = c$. When objective coefficient of forward problem is d^1, optimal solution is $x^1 = (8, 2)$.

Figure : Iteration 1, d^1 and x^1

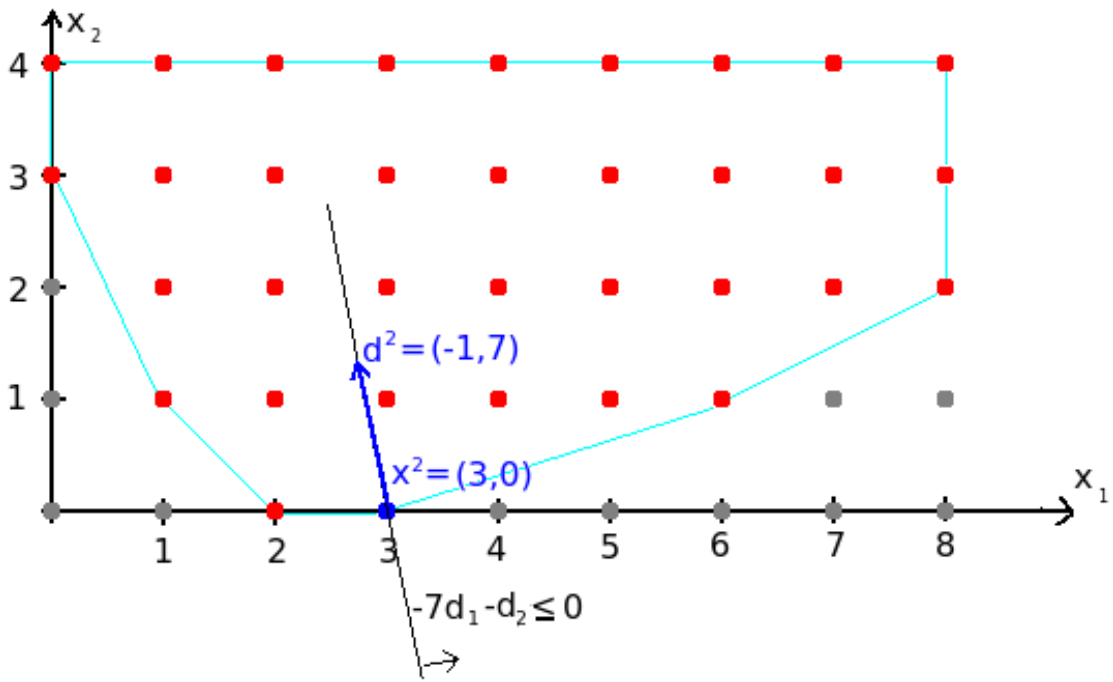


Using x^1, we generate cut $d^\top (x^0 - x^1) \leq 0$, i.e. $-7d_1 - d_2 \leq 0$.

A Small Example: Iteration 2

Following figure shows feasible cone for d. d^2 is the inverse optimal with the current cut. x^2 is the forward optimal solution when d^2 is objective coefficient vector.

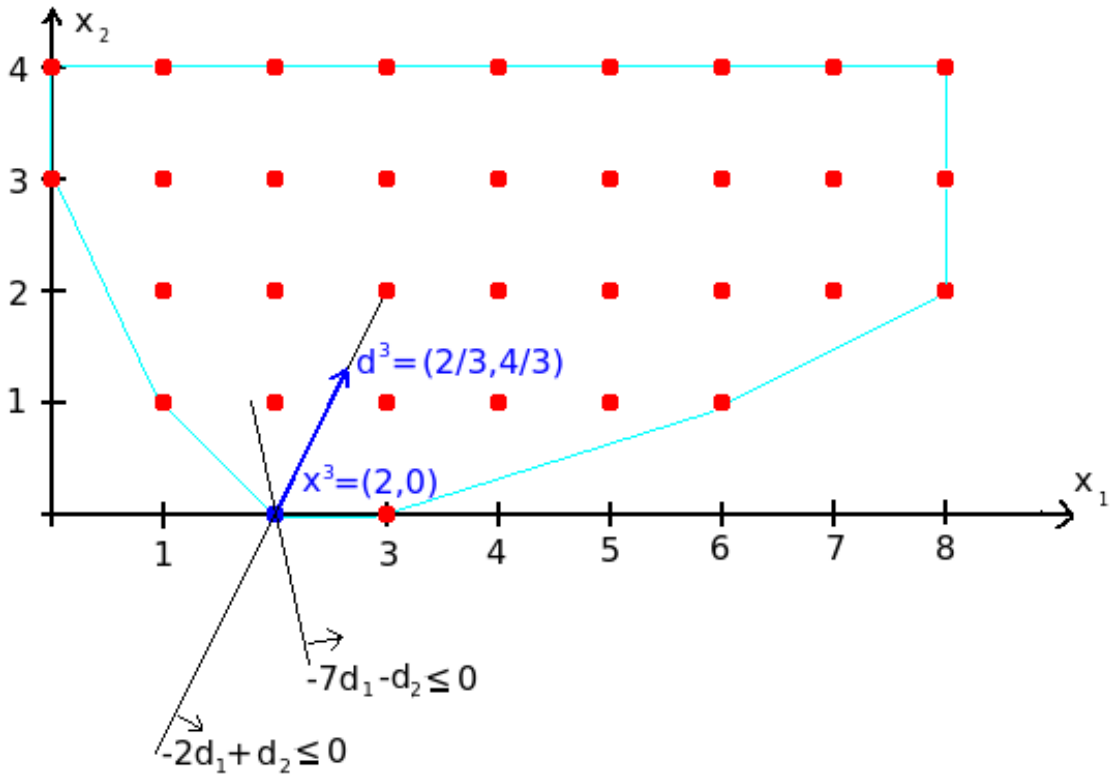
Figure : Iteration 2, feasible d cone ($(3,0)$ is apex), d^2 and x^2



Using x^2, we generate cut $d^\top(x^0 - x^2) \leq 0$, i.e. $-2d_1 + d_2 \leq 0$.

A Small Example: Iteration 3

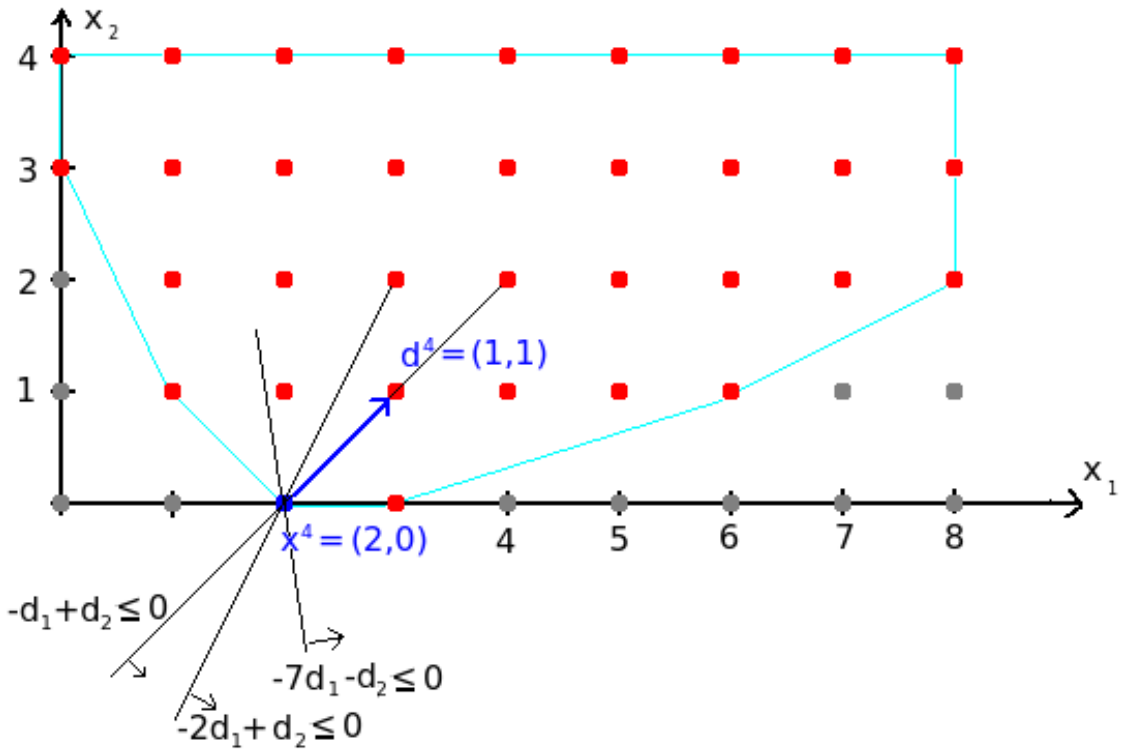
Figure : Iteration 3, feasible d cone, d^3 and x^3



Using x^3, we generate cut $d^\top(x^0 - x^3) \leq 0$, i.e. $-d_1 + d_2 \leq 0$.

A Small Example: Iteration 4

Figure : Iteration 4, feasible d cone, d^4 and x^4



Using x^4, we generate cut $d^\top(x^0 - x^4) \leq 0$, i.e. $-d_1 + d_2 \leq 0$. Note that current d^4 does not violate this cut, then d^4 is the optimal solution of inverse problem.

Outline

Inverse Problems in Practice

- The experiments are designed to test the practical difficulties in solving inverse problems.
- We conjecture that in practice, the difficulties arise from
  - The practical difficulty of solving the forward problem (for a range of objective values)
  - The number of iterations required.
- The number of iterations required is a function of the structure of the forward problem (number of extreme points) and the distance of the optimal objective from the target.
- We therefore conjecture that the practical difficulty of solving these problems will vary substantially betwen problem classes.

Computational Results

- We used MIPLIB3 benchmark problems to create our inverse problems.
- We perturbed objective function coefficients, solved the problems and used the resulting solutions as x^0's.
- Using generated x^0's we created inverse IP problems.
- Experiments are conducted using COIN-OR Branch and Cut (Coin-Cbc) and COIN-OR Open Solver Interface (Coin-Osi) tools with Condor on a cluster running Debian operating system.

Table : MIPLIB Iteration number–l_1

p. name	pert. 0.1	pert. 0.2	pert. 0.3	pert. 0.4	pert. 0.5	pert. 0.6
bell3a	1	1	2	2	11	7
blend2	2	2	2	2	2	2
dcmulti	1	5	1	27	21	47
egout	20	20	20	20	20	20
enigma	1	1	1	1	1	1
flugpl	1	1	1	1	1	1
gen	1	142	188	852	844	1578
gt2	1	1	1	1	1	1
l152lav	16	23	46	46	46	46
lseu	1	1	1	1	1	1
mas74	1	1	1	1	1	1
misc03	1	1	1	1	1	1
misc06	1	1	1	1	1	1
mod008	3	3	6	6	6	6
mod010	51	67	311	390	390	390
modglob	1	1	1	1	1	1
p0201	1	1	4	4	4	4
p0282	1	1	1	1	1	1
pk1	1	1	1	1	1	1
pp08aCUTS	11	12	4	4	14	3
qiu	1	16	10	10	10	10
rgn	1	1	1	1	1	1
rout	1	1	1	1	1	1
stein27	1	1	1	1	1	1
vpm1	1	1	1	1	1	1

Table : MIPLIB Iteration number–$l_{\inf}$

p. name	pert. 0.1	pert. 0.2	pert. 0.3	pert. 0.4	pert. 0.5	pert. 0.6
air03	2	3	2	2	2	2
blend2	3	3	3	2	3	2
cap6000	6	27	30	30	30	32
dcmulti	1	3	1	10	17	21
egout	8	8	8	9	9	9
enigma	1	1	1	1	1	1
flugpl	1	1	1	1	1	1
gen	1	21	35	37	220	179
gt2	1	1	1	1	1	1
harp2	6	6	6	6	7	8
l152lav	4	3	4	4	4	4
lseu	1	1	1	1	1	1
mas74	1	1	1	1	1	1
misc03	1	1	1	1	1	1
misc06	1	1	1	1	1	1
mod008	2	2	3	3	3	2
mod010	4	3	3	3	3	3
modglob	1	1	1	1	1	1
p0201	1	1	2	3	3	3
p0282	1	1	1	1	1	1
pk1	1	1	1	1	1	1
pp08aCUTS	11	11	5	5	8	5
qnet1	1	1	1	1	1	1
rgn	1	1	1	1	1	1
rout	1	1	1	1	1	1
stein27	1	1	1	1	1	1
vpm1	1	1	1	1	1	1

Table : TSPLIB Iteration number–l_1

p. name	pert. 0.1	pert. 0.2	pert. 0.3	pert. 0.4	pert. 0.5	pert. 0.6
att48	1	1	3	4	9	24
berlin52	1	1	1	1	1	1
bier127	73	103	186	307	298	358
burma14	1	1	1	1	1	1
ch130	1	14	31	52	88	81
ch150	1	7	15	20	49	49
eil101	1	4	7	4	8	6
eil51	1	2	4	4	7	16
eil76	1	3	2	4	4	6
kroA100	3	3	6	10	17	38
kroC100	1	3	4	4	8	14
kroD100	1	4	13	24	43	138
kroE100	1	13	13	17	27	26
lin105	1	2	6	18	20	20
pr107	2	13	23	30	38	39
pr124	2	3	6	8	15	13
pr152	3	41	18	18	24	45
rat99	1	10	4	13	21	22
rd100	1	3	8	8	11	11
st70	1	4	13	14	11	14
u159	1	1	1	8	9	13
ulysses16	1	1	1	1	1	1
ulysses22	1	1	3	3	3	5

Table : TSPLIB Iteration number–$l_{\inf}$

p. name	pert. 0.1	pert. 0.2	pert. 0.3	pert. 0.4	pert. 0.5	pert. 0.6
att48	1	1	2	2	3	5
berlin52	1	1	1	1	1	1
bier127	6	5	6	5	6	6
burma14	1	1	1	1	1	1
ch130	1	5	4	6	6	6
ch150	1	4	4	5	5	5
eil101	1	2	2	2	2	2
eil51	1	2	2	2	2	3
eil76	1	3	4	3	3	2
kroA100	2	2	2	3	3	3
kroA150	4	3	5	5	5	5
kroB200	3	4	3	5	6	6
kroC100	1	3	3	3	2	3
kroD100	1	2	4	4	4	5
kroE100	1	5	5	5	5	4
lin105	1	1	3	2	4	4
pr107	1	4	4	6	4	7
pr124	2	3	2	3	3	3
pr136	2	3	4	6	6	6
pr144	3	4	5	5	6	4
pr152	2	3	5	5	3	3
pr76	1	2	3	3	3	3
rat99	1	3	2	4	4	4
rd100	1	2	3	3	3	3
st70	1	2	4	5	4	5
u159	1	1	2	3	3	4
ulysses16	1	1	1	1	1	1
ulysses22	1	1	3	3	2	5

References


Ravindra K. Ahuja and James B. Orlin.
Inverse optimization.
Operations Research, 49(5):771–783, September/October 2001.


Martin Grötschel, László Lovász, and Alexander Schrijver.
Geometric Algorithms and Combinatorial Optimization, volume 2 of *Algorithms and Combinatorics*.
Springer, second corrected edition edition, 1993.


Larry J. Stockmeyer.
The polynomial-time hierarchy.
Theor. Comput. Sci., 3(1):1–22, 1976.

End of presentation

This is end of presentation!

Thank you for listening!