Inverse Optimization

Piotr Wojciechowski[1]

[1] Lane Department of Computer Science and Electrical Engineering
West Virginia University

Outline

Outline

Outline

Outline

1. Definition of Inverse Optimization
   - General Definition
   - Inverse Optimization of Integer Programming
2. Applications of Inverse Optimization
3. Complexity of Inverse Opimization
   - The polynomial Hierarchy
   - Inverse Optimization and the Polynomial Heirarchy

Optimization

In a regular optimization problem we are givien a solution space S and objective function $f : S \to \mathbb{R}$. Our goal is to find $x_0 \in S$ such that $\forall x \in S$ $f(x_0) \leq f(x)$.

Inverse Optimization

In an inverse optimization problem we are givien a solution space S, a target objective function $f : S \to \mathbb{R}$, and a point $x_0 \in S$. Our goal is to find a new objctive function $f' : S \to \mathbb{R}$ such that $\forall x \in S$ $f(x_0) \leq f(x)$ and such that the differecne between f and f' is minimized.

Note

The last requirement is needed to prevent trivial answers such as making f' return 0 for all $x \in S$.

Optimization

In a regular optimization problem we are givien a solution space S and objective function $f : S \to \mathbb{R}$. Our goal is to find $x_0 \in S$ such that $\forall x \in S$ $f(x_0) \leq f(x)$.

Inverse Optimization

In an inverse optimization problem we are givien a solution space S, a target objective function $f : S \to \mathbb{R}$, and a point $x_0 \in S$. Our goal is to find a new objctive function $f' : S \to \mathbb{R}$ such that $\forall x \in S$ $f(x_0) \leq f(x)$ and such that the differecne between f and f' is minimized.

Note

The last requirement is needed to prevent trivial answers such as making f' return 0 for all $x \in S$.

Optimization

In a regular optimization problem we are givien a solution space S and objective function $f : S \to \mathbb{R}$. Our goal is to find $x_0 \in S$ such that $\forall x \in S\ f(x_0) \leq f(x)$.

Inverse Optimization

In an inverse optimization problem we are givien a solution space S, a target objective function $f : S \to \mathbb{R}$, and a point $x_0 \in S$. Our goal is to find a new objctive function $f' : S \to \mathbb{R}$ such that $\forall x \in S\ f(x_0) \leq f(x)$ and such that the differecne between f and f' is minimized.

Note

The last requirement is needed to prevent trivial answers such as making f' return 0 for all $x \in S$.

Optimization

In a regular optimization problem we are givien a solution space S and objective function $f : S \to \mathbb{R}$. Our goal is to find $x_0 \in S$ such that $\forall x \in S\ f(x_0) \leq f(x)$.

Inverse Optimization

In an inverse optimization problem we are givien a solution space S, a target objective function $f : S \to \mathbb{R}$, and a point $x_0 \in S$. Our goal is to find a new objctive function $f' : S \to \mathbb{R}$ such that $\forall x \in S\ f(x_0) \leq f(x)$ and such that the differecne between f and f' is minimized.

Note

The last requirement is needed to prevent trivial answers such as making f' return 0 for all $x \in S$.

Optimization

In a regular optimization problem we are givien a solution space S and objective function $f : S \to \mathbb{R}$. Our goal is to find $x_0 \in S$ such that $\forall x \in S \; f(x_0) \leq f(x)$.

Inverse Optimization

In an inverse optimization problem we are givien a solution space S, a target objective function $f : S \to \mathbb{R}$, and a point $x_0 \in S$. Our goal is to find a new objctive function $f' : S \to \mathbb{R}$ such that $\forall x \in S \; f(x_0) \leq f(x)$ and such that the differecne between f and f' is minimized.

Note

The last requirement is needed to prevent trivial answers such as making f' return 0 for all $x \in S$.

Outline

Definition (Integer Program)

In integer programming our solution space S is a subset of $\mathbb{Z}^n$ defined to be the set of all vectors $\mathbf{x}$ such that $\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$.

Our opimization function $f : \mathbb{Z}^n \to \mathbb{R}$ is defined so that $f(\mathbf{x}) = \mathbf{c} \cdot \mathbf{x}$ for vector $\mathbf{c}$. This problem can be written as

$$\min \mathbf{c} \cdot \mathbf{x}$$
$$\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$$

Definition (Integer Program)

In integer programming our solution space S is a subset of $\mathbb{Z}^n$ defined to be the set of all vectors $\mathbf{x}$ such that $\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$.
Our opimization function $f : \mathbb{Z}^n \rightarrow \mathbb{R}$ is defined so that $f(\mathbf{x}) = \mathbf{c} \cdot \mathbf{x}$ for vector $\mathbf{c}$. This problem can be written as

$$\min \mathbf{c} \cdot \mathbf{x}$$
$$\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$$

Definition (Integer Program)

In integer programming our solution space S is a subset of $\mathbb{Z}^n$ defined to be the set of all vectors $\mathbf{x}$ such that $\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$.
Our opimization function $f : \mathbb{Z}^n \rightarrow \mathbb{R}$ is defined so that $f(\mathbf{x}) = \mathbf{c} \cdot \mathbf{x}$ for vector $\mathbf{c}$. This problem can be written as

$$\min \mathbf{c} \cdot \mathbf{x}$$
$$\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$$

Difference between optimiztion functions

In integer programming each optimization function is defined by a vector **c**. Thus, the difference between two such functions can be defined as the distance between the two corresponding vectors.

There are two such measures or norms that we will consider

1. the l_1 norm, under which $||\mathbf{c} - \mathbf{c}'|| = \sum_{i=1}^{n} |c_i - c_i'|$
2. the l_∞ norm, under which $||\mathbf{c} - \mathbf{c}'|| = \max_{i=1\ldots n} |c_i - c_i'|$

These are the two norms we consider because they can both be expressed using Integer Programs.

Difference between optimiztion functions

In integer programming each optimization function is defined by a vector **c**. Thus, the difference between two such functions can be defined as the distance between the two corresponding vectors.
There are two such measures or norms that we will consider

1. the l_1 norm, under which $||\mathbf{c} - \mathbf{c}'|| = \sum_{i=1}^{n} |c_i - c_i'|$
2. the l_∞ norm, under which $||\mathbf{c} - \mathbf{c}'|| = \max_{i=1\ldots n} |c_i - c_i'|$

These are the two norms we consider because they can both be expressed using Integer Programs.

Difference between optimiztion functions

In integer programming each optimization function is defined by a vector **c**. Thus, the difference between two such functions can be defined as the distance between the two corresponding vectors.

There are two such measures or norms that we will consider

1. the l_1 norm, under which $||\mathbf{c} - \mathbf{c}'|| = \sum_{i=1}^{n} |c_i - c_i'|$
2. the l_∞ norm, under which $||\mathbf{c} - \mathbf{c}'|| = \max_{i=1\dots n} |c_i - c_i'|$

These are the two norms we consider because they can both be expressed using Integer Programs.

Difference between optimiztion functions

In integer programming each optimization function is defined by a vector **c**. Thus, the difference between two such functions can be defined as the distance between the two corresponding vectors.
There are two such measures or norms that we will consider

1. the l_1 norm, under which $||\mathbf{c}-\mathbf{c}'|| = \sum_{i=1}^{n} |c_i - c_i'|$
2. the l_∞ norm, under which $||\mathbf{c}-\mathbf{c}'|| = \max_{i=1\ldots n} |c_i - c_i'|$

These are the two norms we consider because they can both be expressed using Integer Programs.

Difference between optimiztion functions

In integer programming each optimization function is defined by a vector $\mathbf{c}$. Thus, the difference between two such functions can be defined as the distance between the two corresponding vectors.

There are two such measures or norms that we will consider

1. the l_1 norm, under which $||\mathbf{c} - \mathbf{c}'|| = \sum_{i=1}^{n} |c_i - c_i'|$
2. the l_∞ norm, under which $||\mathbf{c} - \mathbf{c}'|| = \max_{i=1\ldots n} |c_i - c_i'|$

These are the two norms we consider because they can both be expressed using Integer Programs.

Definition (Inverse Optimization of an Integer Program)

Given the solution space S and vector $\mathbf{c}$ and $\mathbf{x_0}$, we want to find vector $\mathbf{c}'$ such that $||\mathbf{c} - \mathbf{c}'||$ is minimized and so that $\forall \mathbf{x} \in S$ $\mathbf{c}' \cdot \mathbf{x_0} \leq \mathbf{c}' \cdot \mathbf{x}$ We can simplify this problem by restricting our choices of $\mathbf{x}$ from S to the finite set, E, of extreme points.

l_1 norm

Under the l_1 norm this program can be written as

$$\min \sum_{i=1}^{n} \theta_i$$

$$\begin{aligned} c_i - c'_i &\leq \theta_i : i = 1 \dots n \\ c'_i - c_i &\leq \theta_i : i = 1 \dots n \\ \mathbf{c}' \cdot \mathbf{x_0} &\leq \mathbf{c}' \cdot \mathbf{x} : \forall \mathbf{x} \in E \end{aligned}$$

Definition (Inverse Optimization of an Integer Program)

Given the solution space S and vector $\mathbf{c}$ and $\mathbf{x_0}$, we want to find vector $\mathbf{c}'$ such that $||\mathbf{c} - \mathbf{c}'||$ is minimized and so that $\forall \mathbf{x} \in S \; \mathbf{c}' \cdot \mathbf{x_0} \leq \mathbf{c}' \cdot \mathbf{x}$ We can simplify this problem by restricting our choices of $\mathbf{x}$ from S to the finite set, E, of extreme points.

l_1 norm

Under the l_1 norm this program can be written as

$$\min \sum_{i=1}^{n} \theta_i$$

$$\begin{aligned} c_i - c_i' &\leq \theta_i : i = 1 \ldots n \\ c_i' - c_i &\leq \theta_i : i = 1 \ldots n \\ \mathbf{c}' \cdot \mathbf{x_0} &\leq \mathbf{c}' \cdot \mathbf{x} : \forall \mathbf{x} \in E \end{aligned}$$

Definition (Inverse Optimization of an Integer Program)

Given the solution space S and vector $\mathbf{c}$ and $\mathbf{x_0}$, we want to find vector $\mathbf{c}'$ such that $||\mathbf{c} - \mathbf{c}'||$ is minimized and so that $\forall \mathbf{x} \in S$ $\mathbf{c}' \cdot \mathbf{x_0} \leq \mathbf{c}' \cdot \mathbf{x}$ We can simplify this problem by restricting our choices of $\mathbf{x}$ from S to the finite set, E, of extreme points.

l_1 norm

Under the l_1 norm this program can be written as

$$\min \sum_{i=1}^{n} \theta_i$$

$$\begin{array}{rcl} c_i - c_i' & \leq & \theta_i : \ i = 1 \ldots n \\ c_i' - c_i & \leq & \theta_i : \ i = 1 \ldots n \\ \mathbf{c}' \cdot \mathbf{x_0} & \leq & \mathbf{c}' \cdot \mathbf{x} : \ \forall \mathbf{x} \in E \end{array}$$

l_∞ norm

Under the l_∞ norm this program can be written as

$$\begin{aligned}
&\min \theta \\
c_i - c_i' &\leq \theta : i = 1 \ldots n \\
c_i' - c_i &\leq \theta : i = 1 \ldots n \\
\mathbf{c}' \cdot \mathbf{x_0} &\leq \mathbf{c}' \cdot \mathbf{x} : \forall \mathbf{x} \in E
\end{aligned}$$

Applications

1. Inverse shortest path problems, used in predicting the movement of earthquakes, arising in geophysical sciences.
2. Inverse programs naturally arise in situations when we want to infer a person's utility function from the person's actions.
3. In numerical analysis, errors for solving the system $\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$ are measured in terms of inverse optimization.
4. Any kind of system that we can observe the solutions but can not measure all the parameters that results in these solutions.

Applications

1. Inverse shortest path problems, used in predicting the movement of earthquakes, arising in geophysical sciences.
2. Inverse programs naturally arise in situations when we want to infer a person's utility function from the person's actions.
3. In numerical analysis, errors for solving the system $\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$ are measured in terms of inverse optimization.
4. Any kind of system that we can observe the solutions but can not measure all the parameters that results in these solutions.

Applications

1. Inverse shortest path problems, used in predicting the movement of earthquakes, arising in geophysical sciences.
2. Inverse programs naturally arise in situations when we want to infer a person's utility function from the person's actions.
3. In numerical analysis, errors for solving the system $\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$ are measured in terms of inverse optimization.
4. Any kind of system that we can observe the solutions but can not measure all the parameters that results in these solutions.

Applications

1. Inverse shortest path problems, used in predicting the movement of earthquakes, arising in geophysical sciences.
2. Inverse programs naturally arise in situations when we want to infer a person's utility function from the person's actions.
3. In numerical analysis, errors for solving the system $\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$ are measured in terms of inverse optimization.
4. Any kind of system that we can observe the solutions but can not measure all the parameters that results in these solutions.

Outline

Definition (Polynomial Hierarchy)

The *polynomial hierarchy* is the following sequence of classes:

1. $\Delta_0 P = \Sigma_0 P = \Pi_0 P = P$
2. $\Delta_{i+1} P = P^{\Sigma_i P}$
3. $\Sigma_{i+1} P = NP^{\Sigma_i P}$
4. $\Pi_{i+1} P = coNP^{\Sigma_i P}$

For all $i \geq 0$.
We also define the collective class $PH = \bigcup_{i \geq 0} \Sigma_i P$.

Observations

Note that because $\Sigma_0 P = P$, we have that $\Sigma_1 P = NP$, $\Delta_1 P = P$, and $\Pi_1 P = coNP$. At each level the classes are believed to be distinct and are known to hold the same relationship as P, NP and coNP. Also, each class at each level includes all classes at the previous levels.

Definition (Polynomial Hierarchy)

The *polynomial hierarchy* is the following sequence of classes:

1. $\Delta_0 \mathrm{P} = \Sigma_0 \mathrm{P} = \Pi_0 \mathrm{P} = \mathrm{P}$
2. $\Delta_{i+1} \mathrm{P} = P^{\Sigma_i P}$
3. $\Sigma_{i+1} \mathrm{P} = NP^{\Sigma_i P}$
4. $\Pi_{i+1} \mathrm{P} = coNP^{\Sigma_i P}$

For all $i \geq 0$.
We also define the collective class $PH = \bigcup_{i \geq 0} \Sigma_i P$.

Observations

Note that because $\Sigma_0 \mathrm{P} = \mathrm{P}$, we have that $\Sigma_1 \mathrm{P} = \mathrm{NP}$, $\Delta_1 \mathrm{P} = \mathrm{P}$, and $\Pi_1 \mathrm{P} = \mathrm{coNP}$. At each level the classes are believed to be distinct and are known to hold the same relationship as P, NP and coNP. Also, each class at each level includes all classes at the previous levels.

Definition (Polynomial Hierarchy)

The *polynomial hierarchy* is the following sequence of classes:

1. $\Delta_0\text{P} = \Sigma_0\text{P} = \Pi_0\text{P} = \text{P}$
2. $\Delta_{i+1}\text{P} = P^{\Sigma_i P}$
3. $\Sigma_{i+1}\text{P} = NP^{\Sigma_i P}$
4. $\Pi_{i+1}\text{P} = coNP^{\Sigma_i P}$

For all $i \geq 0$.
We also define the collective class $PH = \bigcup_{i \geq 0} \Sigma_i P$.

Observations

Note that because $\Sigma_0\text{P} = \text{P}$, we have that $\Sigma_1\text{P} = \text{NP}$, $\Delta_1\text{P} = \text{P}$, and $\Pi_1\text{P} = \text{coNP}$. At each level the classes are believed to be distinct and are known to hold the same relationship as P, NP and coNP. Also, each class at each level includes all classes at the previous levels.

Definition (Polynomial Hierarchy)

The *polynomial hierarchy* is the following sequence of classes:

1. $\Delta_0 \mathrm{P} = \Sigma_0 \mathrm{P} = \Pi_0 \mathrm{P} = \mathrm{P}$
2. $\Delta_{i+1} \mathrm{P} = P^{\Sigma_i P}$
3. $\Sigma_{i+1} \mathrm{P} = NP^{\Sigma_i P}$
4. $\Pi_{i+1} \mathrm{P} = coNP^{\Sigma_i P}$

For all $i \geq 0$.
We also define the collective class $PH = \bigcup_{i \geq 0} \Sigma_i P$.

Observations

Note that because $\Sigma_0 \mathrm{P} = \mathrm{P}$, we have that $\Sigma_1 \mathrm{P} = \mathrm{NP}$, $\Delta_1 \mathrm{P} = \mathrm{P}$, and $\Pi_1 \mathrm{P} = \mathrm{coNP}$. At each level the classes are believed to be distinct and are known to hold the same relationship as P, NP and coNP. Also, each class at each level includes all classes at the previous levels.

Definition (Polynomial Hierarchy)

The *polynomial hierarchy* is the following sequence of classes:

1. $\Delta_0 \mathrm{P} = \Sigma_0 \mathrm{P} = \Pi_0 \mathrm{P} = \mathrm{P}$
2. $\Delta_{i+1} \mathrm{P} = P^{\Sigma_i P}$
3. $\Sigma_{i+1} \mathrm{P} = NP^{\Sigma_i P}$
4. $\Pi_{i+1} \mathrm{P} = coNP^{\Sigma_i P}$

For all $i \geq 0$.
We also define the collective class $PH = \bigcup_{i \geq 0} \Sigma_i P$.

Observations

Note that because $\Sigma_0 \mathrm{P} = \mathrm{P}$, we have that $\Sigma_1 \mathrm{P} = \mathrm{NP}$, $\Delta_1 \mathrm{P} = \mathrm{P}$, and $\Pi_1 \mathrm{P} = \mathrm{coNP}$. At each level the classes are believed to be distinct and are known to hold the same relationship as P, NP and coNP. Also, each class at each level includes all classes at the previous levels.

Definition (Polynomial Hierarchy)

The *polynomial hierarchy* is the following sequence of classes:

1. $\Delta_0\mathrm{P} = \Sigma_0\mathrm{P} = \Pi_0\mathrm{P} = \mathrm{P}$
2. $\Delta_{i+1}\mathrm{P} = P^{\Sigma_i P}$
3. $\Sigma_{i+1}\mathrm{P} = NP^{\Sigma_i P}$
4. $\Pi_{i+1}\mathrm{P} = coNP^{\Sigma_i P}$

For all $i \geq 0$.

We also define the collective class $PH = \bigcup_{i \geq 0} \Sigma_i P$.

Observations

Note that because $\Sigma_0\mathrm{P} = \mathrm{P}$, we have that $\Sigma_1\mathrm{P} = \mathrm{NP}$, $\Delta_1\mathrm{P} = \mathrm{P}$, and $\Pi_1\mathrm{P} = \mathrm{coNP}$. At each level the classes are believed to be distinct and are known to hold the same relationship as P, NP and coNP. Also, each class at each level includes all classes at the previous levels.

Definition (Polynomial Hierarchy)

The *polynomial hierarchy* is the following sequence of classes:

1. $\Delta_0\mathrm{P} = \Sigma_0\mathrm{P} = \Pi_0\mathrm{P} = \mathrm{P}$
2. $\Delta_{i+1}\mathrm{P} = P^{\Sigma_i P}$
3. $\Sigma_{i+1}\mathrm{P} = NP^{\Sigma_i P}$
4. $\Pi_{i+1}\mathrm{P} = coNP^{\Sigma_i P}$

For all $i \geq 0$.
We also define the collective class $PH = \bigcup_{i \geq 0} \Sigma_i P$.

Observations

Note that because $\Sigma_0\mathrm{P} = \mathrm{P}$, we have that $\Sigma_1\mathrm{P} = \mathrm{NP}$, $\Delta_1\mathrm{P} = \mathrm{P}$, and $\Pi_1\mathrm{P} = \mathrm{coNP}$. At each level the classes are believed to be distinct and are known to hold the same relationship as P, NP and coNP. Also, each class at each level includes all classes at the previous levels.

Definition (Polynomial Hierarchy)

The *polynomial hierarchy* is the following sequence of classes:

1. $\Delta_0 \mathrm{P} = \Sigma_0 \mathrm{P} = \Pi_0 \mathrm{P} = \mathrm{P}$
2. $\Delta_{i+1} \mathrm{P} = P^{\Sigma_i P}$
3. $\Sigma_{i+1} \mathrm{P} = NP^{\Sigma_i P}$
4. $\Pi_{i+1} \mathrm{P} = coNP^{\Sigma_i P}$

For all $i \geq 0$.
We also define the collective class $PH = \bigcup_{i \geq 0} \Sigma_i P$.

Observations

Note that because $\Sigma_0 \mathrm{P} = \mathrm{P}$, we have that $\Sigma_1 \mathrm{P} = \mathrm{NP}$, $\Delta_1 \mathrm{P} = \mathrm{P}$, and $\Pi_1 \mathrm{P} = \mathrm{coNP}$. At each level the classes are believed to be distinct and are known to hold the same relationship as P, NP and coNP. Also, each class at each level includes all classes at the previous levels.

Definition (Polynomial Hierarchy)

The *polynomial hierarchy* is the following sequence of classes:

1. $\Delta_0\text{P} = \Sigma_0\text{P} = \Pi_0\text{P} = \text{P}$
2. $\Delta_{i+1}\text{P} = P^{\Sigma_i P}$
3. $\Sigma_{i+1}\text{P} = NP^{\Sigma_i P}$
4. $\Pi_{i+1}\text{P} = coNP^{\Sigma_i P}$

For all $i \geq 0$.
We also define the collective class $PH = \bigcup_{i \geq 0} \Sigma_i P$.

Observations

Note that because $\Sigma_0\text{P} = \text{P}$, we have that $\Sigma_1\text{P} = \text{NP}$, $\Delta_1\text{P} = \text{P}$, and $\Pi_1\text{P} = \text{coNP}$. At each level the classes are believed to be distinct and are known to hold the same relationship as P, NP and coNP. Also, each class at each level includes all classes at the previous levels.

Definition (Polynomial Hierarchy)

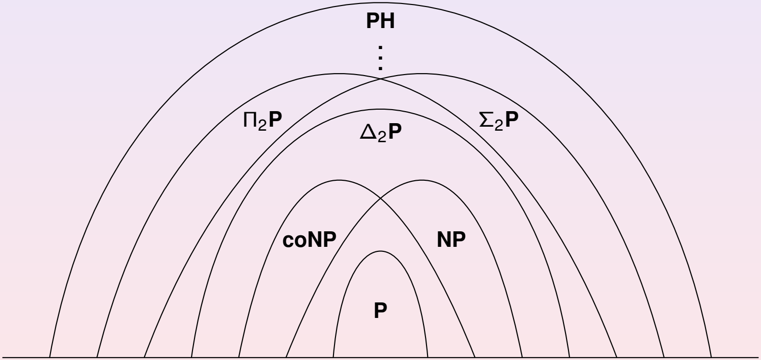
The *polynomial hierarchy* is the following sequence of classes:

1. $\Delta_0 P = \Sigma_0 P = \Pi_0 P = P$
2. $\Delta_{i+1} P = P^{\Sigma_i P}$
3. $\Sigma_{i+1} P = NP^{\Sigma_i P}$
4. $\Pi_{i+1} P = coNP^{\Sigma_i P}$

For all $i \geq 0$.
We also define the collective class $PH = \bigcup_{i \geq 0} \Sigma_i P$.

Observations

Note that because $\Sigma_0 P = P$, we have that $\Sigma_1 P = NP$, $\Delta_1 P = P$, and $\Pi_1 P = coNP$. At each level the classes are believed to be distinct and are known to hold the same relationship as P, NP and coNP. Also, each class at each level includes all classes at the previous levels.


PH
⋮
Π₂P
Δ₂P
Σ₂P
coNP
NP
P

Outline

1. Definition of Inverse Optimization
   - General Definition
   - Inverse Optimization of Integer Programming
2. Applications of Inverse Optimization
3. Complexity of Inverse Opimization
   - The polynomial Hierarchy
   - Inverse Optimization and the Polynomial Heirarchy

Theorem

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

Theorem

Inverse IP optimization problem under either the l_1 or l_∞ norm is solvable in polynomial time when using an IP oracle

Theorem

The inverse IP decision problem is in Δ_2^P.

Conjecture

It is still unkown if the inverse IP decision problem is Δ_2^P-Complete.

Theorem

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

Theorem

Inverse IP optimization problem under either the l_1 or l_∞ norm is solvable in polynomial time when using an IP oracle

Theorem

The inverse IP decision problem is in Δ_2^P.

Conjecture

It is still unkown if the inverse IP decision problem is Δ_2^P-Complete.

Theorem

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

Theorem

Inverse IP optimization problem under either the l_1 or l_∞ norm is solvable in polynomial time when using an IP oracle

Theorem

The inverse IP decision problem is in Δ_2^P.

Conjecture

It is still unkown if the inverse IP decision problem is Δ_2^P-Complete.

Theorem

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

Theorem

Inverse IP optimization problem under either the l_1 or l_∞ norm is solvable in polynomial time when using an IP oracle

Theorem

The inverse IP decision problem is in Δ_2^P.

Conjecture

It is still unkown if the inverse IP decision problem is Δ_2^P-Complete.