Inverse Optimization

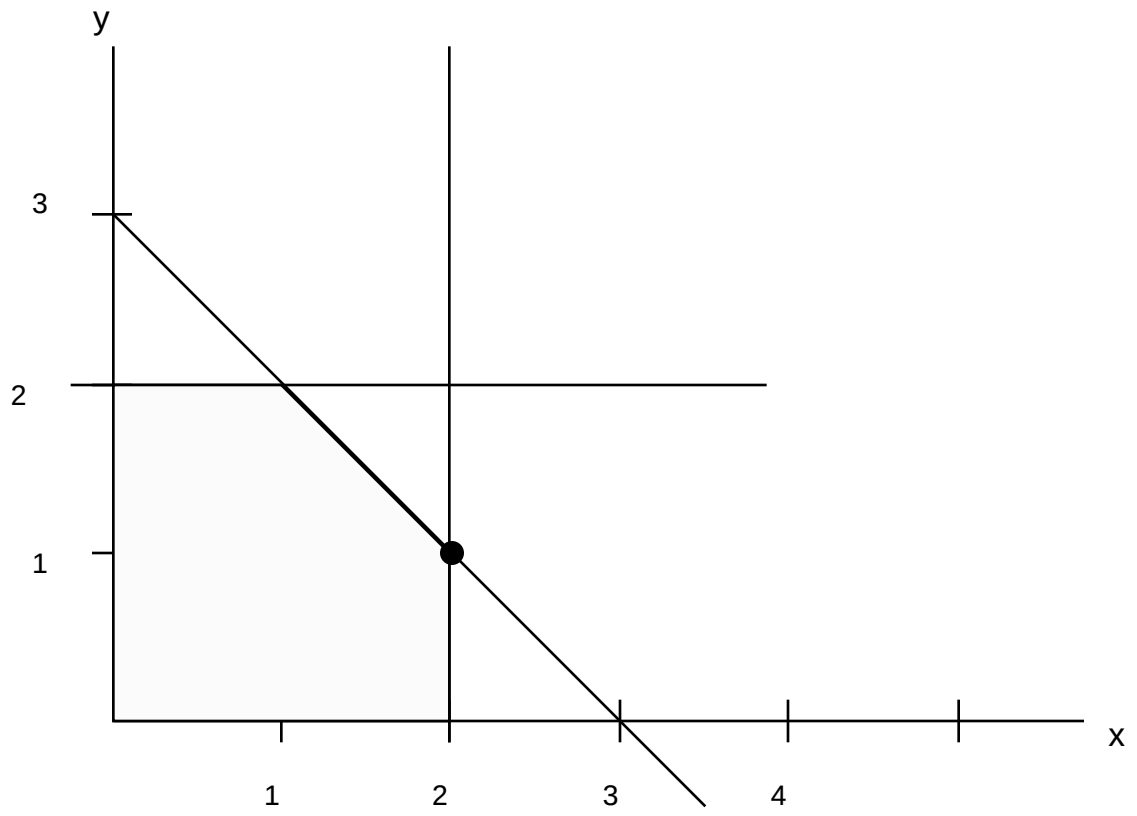
by James Orlin

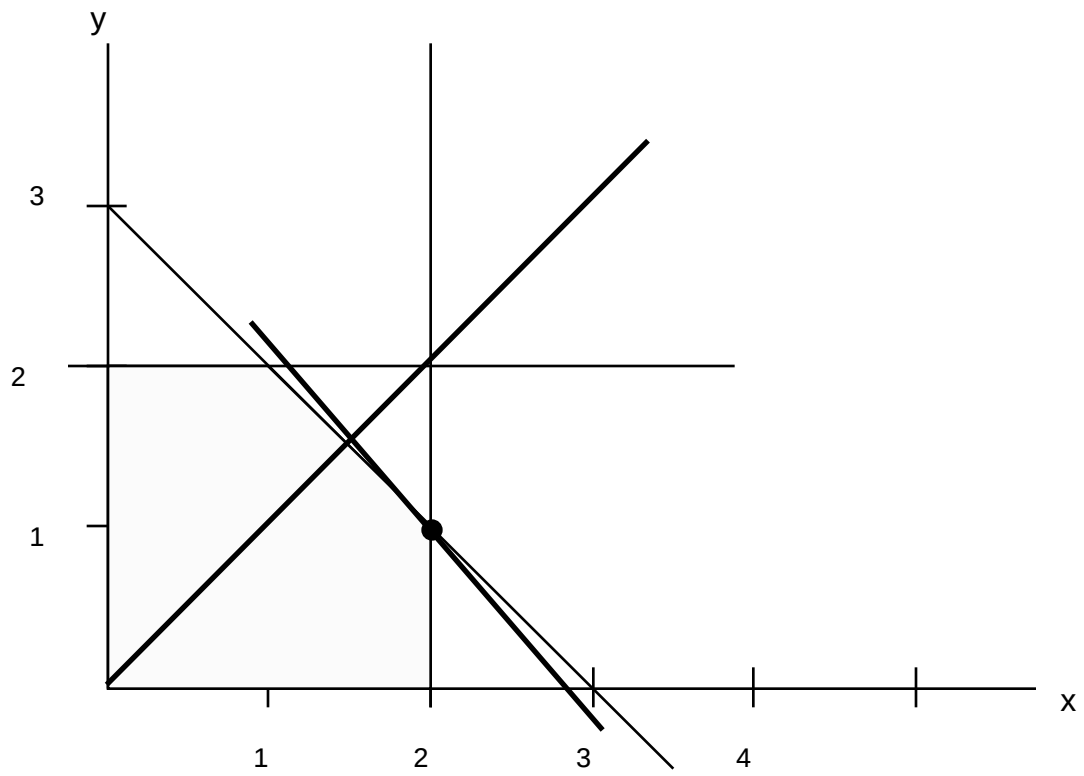
based on research that is joint with Ravi Ahuja

Jeopardy 2000 -- the Math Programming Edition

The category is “linear objective functions”

The answer: When you maximize this linear objective function, over the feasible region given below, the optimal answer is x* = (2,1).

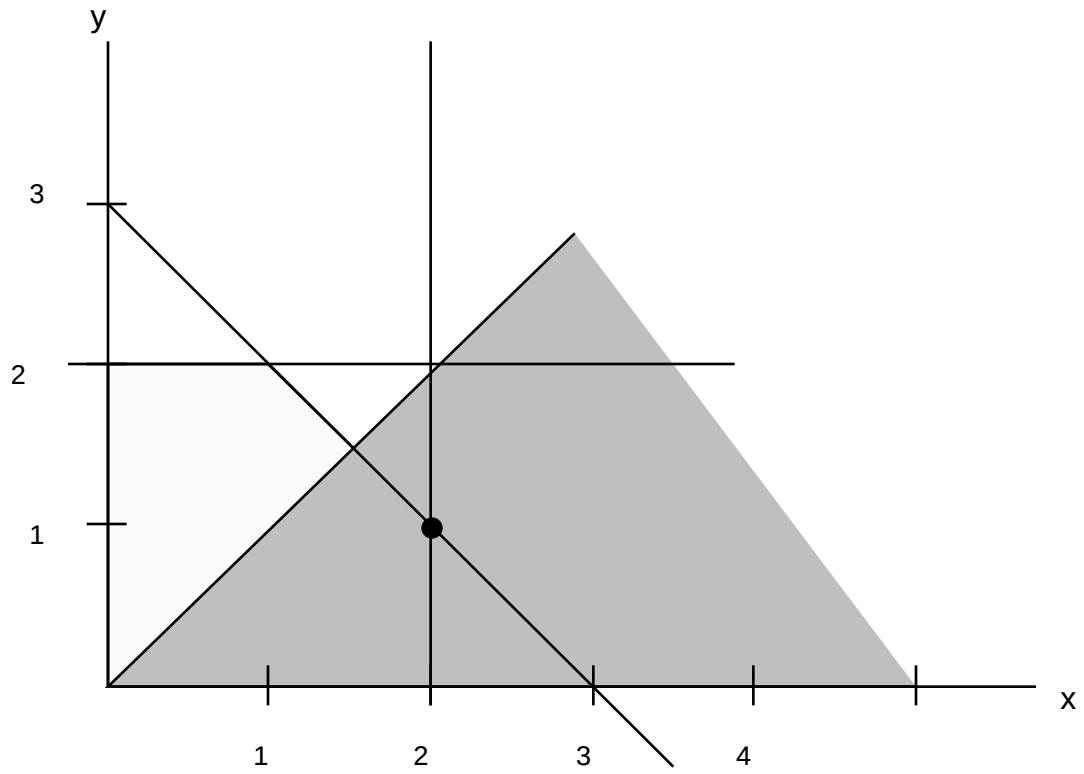




The objective could be x + y or 2x + 2y or 3x + 3y or ....
The objective could be x or 2x or 3x or ...

or it could be any non-negative combination of x+y and x.

Notes: For this problem the feasible region for the inverse problem is a polyhedral cone.
The cone depends only on the active constraints.



We will define inverse problems for minimization.
If the original problem is: minimize (cx : x P), and if x* is a proposed solution, then the feasible region for the *inverse problem* is Inv(x*,P) = {d : x* optimizes the following math program: minimize (dx : x P).

Jeopardy 2000 version.

Answer. The solution x* minimizes dx : x P for this linear objective d.

Defining the inverse problem.

Original problem:

Minimize (cx : x P).

Inverse problem, given a solution x* P:

Minimize (|| d - c || : d Inv(x*, P))

that is x* is optimal for minimize (dx : x P).

For most of this talk, we assume that $|| d - c || = \sum_i | c_i - d_i |$.

Why study inverse problems?

1. For some algorithms, such as cost scaling algorithms for minimum cost flows, the solutions converge to the optimal in an inverse manner. (A solution is ε-optimal at the ε-scaling phase, and ε continuously decreases towards 0.)

2. For problems in which the cost coefficients are not known with precision, it is plausible to call a solution x* for min (cx : x P) nearly optimal if there is some nearby cost vector c' such that x* is optimal for min (c'x : x P).

3. In numerical analysis, errors for solving the system Ax = b are measured in terms of inverse optimization.

4. It might lead to some interesting theory for measuring errors for NP-complete problems.

5. There are some interesting occasions in which solving the inverse problem is faster than solving for the optimum solution. (An example is matroid intersection for representable matroids.)

Inverse problems when the original problem is an LP.

Original problem

Minimize $(cx : Ax = b, x \geq 0)$.
$P = \{x : Ax = b, x \geq 0\}$.

$Inv(x^*, P) = \{d : dx^* = \pi b, \pi A \leq d\}$.

So the inverse problem is as follows:

Minimize $\sum_i | c_i - d_i |$.

subject to $\pi A \leq d$
$\pi b - dx^* = 0$.

Inverse problems in general

Theorem. Suppose that min (dx : x P) is solvable in polynomial time for each vector d. Then then the inverse problem defined for min (cx : x P) is solvable in polynomial time for all vectors c.

Proof.
For a given solution x*,

Inv(x*,P) = { d : dx* ≤ dx' for all x' P).

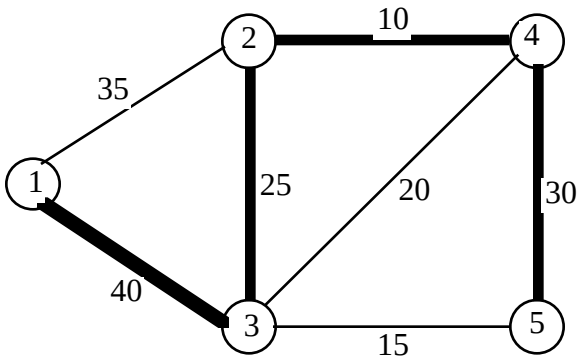
So the inverse problem for min (cx : x P) wrt solution x* is:

Minimize $\sum_i | c_i - d_i |$.
subject to dx* - dx' ≤ 0 for all x' P.

Claim: this problem is solvable in polynomial time using the ellipsoid algorithm.

Proof of claim. A proposed vector d is infeasible for the above LP if there is some x' P so that dx* - dx' > 0.
To find x' P, let x' so that dx' = min(dx : x P).

The Minimum Spanning Tree Problem.



T* = { (1,3), (2,3), (2,4), (4,5) }.

Let δ_{ij} be the amount added to the cost of arc (i,j)

Note: $\delta_{ij} \leq 0$ for (i,j) T*; $\delta_{ij} \geq 0$ for (i,j) T*;

The inverse optimization problem:

Minimize $\sum_{(i,j)\ T^*} - \delta_{ij} \quad + \quad \sum_{(i,j)\ T^*} \delta_{ij}$

subject to

$35 + \delta_{12} \geq 40 + \delta_{13}$

$35 + \delta_{12} \geq 25 + \delta_{23}$

$20 + \delta_{34} \geq 25 + \delta_{23}$

$20 + \delta_{34} \geq 10 + \delta_{24}$

$15 + \delta_{35} \geq 10 + \delta_{24}$

$15 + \delta_{35} \geq 25 + \delta_{23}$

$15 + \delta_{35} \geq 30 + \delta_{45}$

Note that this is the dual of a network flow problem.

Running times for the minimum spanning tree problem.

Min cost flow problem with m nodes, and at most nm arcs.

Running time $O(nm^2 \log n \log nC)$

Assignment problem with n-1 nodes on one side, m-n+1 nodes on the other side, and nm arcs.

Running time $O(n^2m)$

Speedup of the assignment problem:

Running time $O(n^3)$

Further speedup of the assignment problem:

Running time $O(n^2 \log n)$.

Generalization to matroids and matroid intersection.

Solving the inverse problem without solving the dual: Version 1

Original Problem: Maximize (cy : y P),
where P = {y : Ay ≤ 0, y ≥ 0}.
Proposed solution y* = 0;

Inverse problem: Minimize || d - c || : d Inv(0, P).

Theorem. The inverse problem can be solved by solving:

Maximize cy
subject to Ay ≤ 0
0 ≤ y ≤ 1.

Let $F(\alpha) = \sum_j \alpha_j$
The inverse problem may be restated as follows:

minimize F(α)
subject toc - α Inv(0, P)

Proof. Consider the lagrangian relaxation formed by relaxing the upper and lower bound constraints.

$$L(\alpha) = \quad \text{Max} \sum c_j y_j + \sum \alpha_j (1 - y_j)$$
$$\text{s.t} \quad Ay \le 0$$
$$y \ge 0$$

$$\text{Max} \sum \alpha_j + \sum (c_j - \alpha_j) y_j$$
$$\text{s.t} \quad Ay \le 0$$
$$y \ge 0$$

So, $L(\alpha) = \sum \alpha_j$ if $y^* = 0$ is optimal for

Maximize $((c - \alpha) : x \quad P)$

$= \infty$ otherwise;

That is $L(\alpha) = F(\alpha)$ if $(c-\alpha) \quad Inv(0,P)$

$= \infty$ otherwise.

The optimal solution to the lagrangian dual is

$L^* = \min [L(\alpha) : \alpha \ge 0] = \min \{F(\alpha) : \alpha \quad Inv(0,P)$.

Lemma. The optimal solution for the lagrangian dual yields the optimal objective function for the inverse optimization problem.

Solving the inverse problem without solving the dual. Version 2

Original Problem: Minimize $(cx : x \quad P)$,
where $P = \{x : Ax \leq b, \quad A'x \leq b', \quad x \geq 0\}$.
Active *Inactive*

Proposed solution x*.
Suppose that $Ax^* = b$, $A'x^* < b'$, and $x^*_j = 0$ for $j \quad S$;

Inverse problem: Minimize $\| d - c \| : d \quad Inv(x^*, P)$.

Theorem. The inverse problem can be solved by solving:

$$\begin{aligned} &\text{Minimize} && cx \\ &\text{subject to} && Ax \leq b \\ & && 0 \leq x_j \leq 1 \quad \text{for } j \quad S \\ & && x^*_j - 1 \leq x_j \leq x^*_j + 1 \quad \text{for } j \quad S \end{aligned}$$

Proof. A reduction to the previous Theorem.

Application to convex hulls of 0-1 integer programs.

original problem

Minimize	cx		
subject to	$Ax \leq b$		*Active*
	$A'x \leq b'$		*Inactive*
	$0 \leq x \leq 1$		

Suppose that $Ax^* = b$ and $A'x^* < b'$, and $x^*_j = 0$ or 1 for each j.

Theorem. The inverse optimization problem is equivalent to

minimize	cx		
subject to	$Ax \leq$	b	
	$0 \leq x \leq$	1	

Proof. If $x^*_j = 0$, we keep the binding constraint $x_j \geq 0$ in the inverse problem and introduce the constraint $x^*_j \leq 1$ to the inverse. If $x^*_j = 1$, then we keep the binding constraint $x_j \leq 1$ in the inverse problem, and introduce the constraint $x^*_j \geq 0$.

The optimal objective value for this inverse problem is

minimum $\sum \alpha_j$
subject toc-α Inv(x*, P).

Let z_{inv} = optimal objective value for the inverse problem.
Let z_{opt} = optimal objective value for the original problem.

Let Dev = $cx^* - z_{opt}$.

Corollary. For 0-1 linear programs, if x* is integral, then

$$z_{inv} \geq \text{Dev}.$$

Moreover, z_{inv} = Dev if each non 0-1 constraint is binding.

Application to Assignment Problem.

Original Problem

Minimize $\sum_{i,j} c_{ij} x_{ij}$

Subject to $\sum_{j} x_{ij} = 1$ for each i

$\sum_{i} x_{ij} = 1$ for each j

$0 \leq x_{ij} \leq 1$

Inverse Problem (same as the original)

Minimize $\sum_{i,j} c_{ij} x_{ij}$

Subject to $\sum_{j} x_{ij} = 1$ for each i

$\sum_{i} x_{ij} = 1$ for each j

$0 \leq x_{ij} \leq 1$

Goal: find a maximum weight assignment

	APL	Basic	COBOL	D	Excel	Fortran
Albert	13	13	11	10	10	13
Ben	18	15	17	17	19	10
Caroline	10	16	15	19	10	18
Donna	17	17	14	15	11	16
Eduardo	11	17	16	14	18	17
Frank	16	18	10	19	13	15

CurrentObjective = 91 BestObjective = 102

	-1	-1	-2	0	0	-1
14	0	0	-1	-4	-4	0
19	0	-3	0	-2	0	-8
19	-8	-2	-2	0	-9	0
18	0	0	-2	-3	-7	-1
18	-6	0	0	-4	0	0
19	-2	0	-7	0	-6	-3

CurrentObjective = -11 BestObjective = 0

Application to the shortest path problem single source (node 1), single sink (node n)

Original Problem

Minimize $\sum_{i,j} c_{ij} x_{ij}$

Subject to $\sum_j x_{ij} - \sum_j x_{ji} = \begin{cases} 1 & \text{if } i = 1 \\ -1 & \text{if } i = n \\ 0 & \text{otherwise} \end{cases}$

$$0 \le x_{ij} \le 1$$

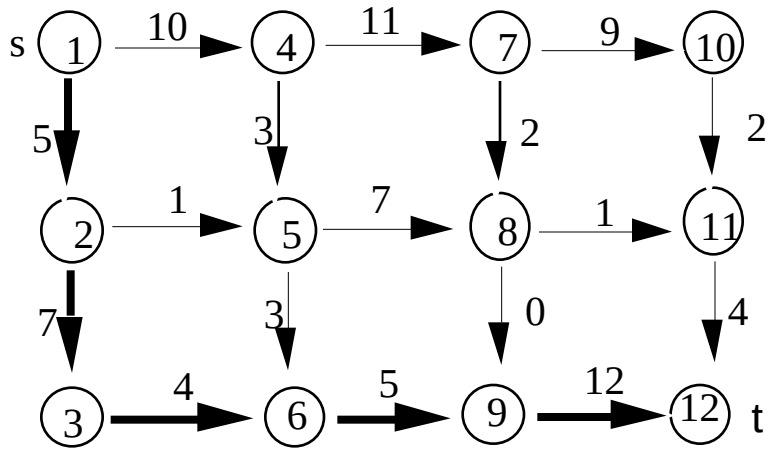
Inverse Problem (same as the original problem)

Minimize $\sum_{i,j} c_{ij} x_{ij}$

Subject to $\sum_j x_{ij} - \sum_j x_{ji} = \begin{cases} 1 & \text{if } i = 1 \\ -1 & \text{if } i = n \\ 0 & \text{otherwise} \end{cases}$

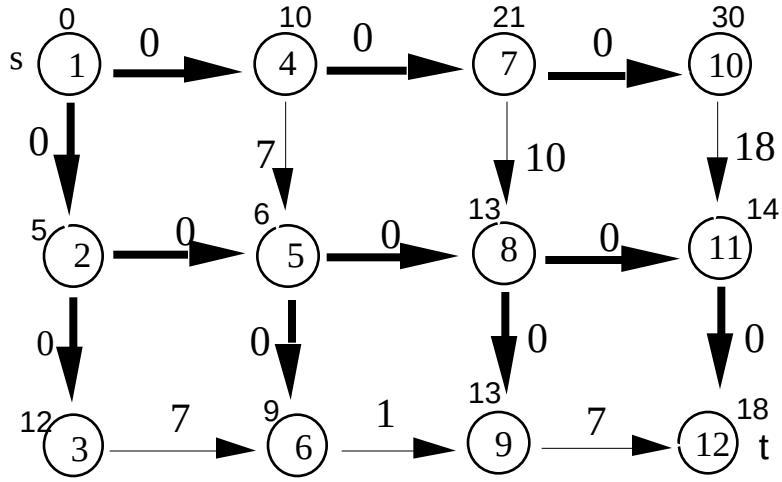
$$0 \le x_{ij} \le 1$$

Example. Network with proposed path.



Here is the proposed shortest path.

It's length is 33.



Here is the shortest path tree.

The length of the shortest path from s to t is only 18.

The arc values are reduced costs.

Application to Minimum Cost Flow

Assume without loss of generality that we are looking for a non-negative circulation, and that $x^* = 0$;

Original Problem

Minimize $\sum_{i,j} c_{ij} x_{ij}$

Subject to $\sum_j x_{ij} - \sum_j x_{ji} = 0$ for each i

$0 \leq x_{ij} \leq u_{ij}$

Inverse Problem

Minimize $\sum_{i,j} c_{ij} x_{ij}$

Subject to $\sum_j x_{ij} - \sum_j x_{ji} = 0$ for each i

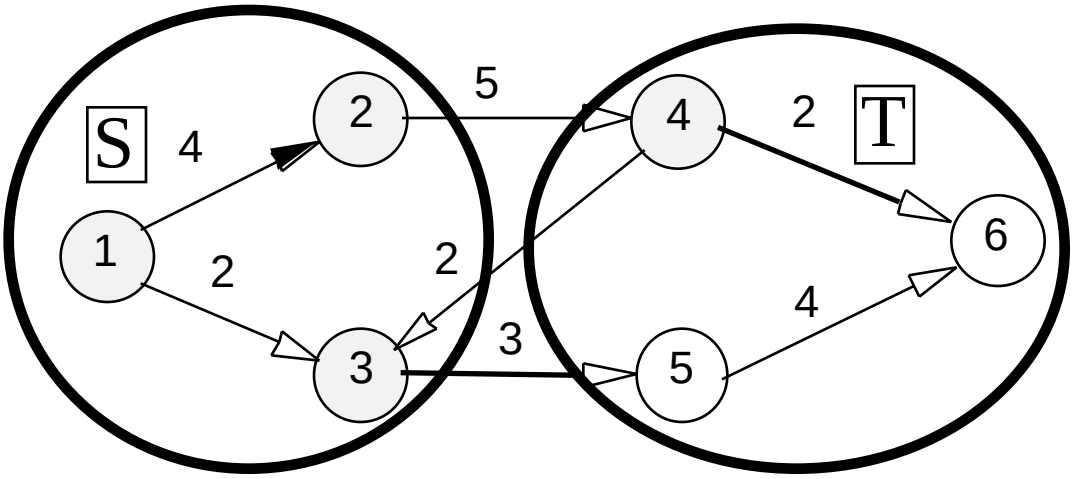
$0 \leq x_{ij} \leq 1$

Application to the Minimum s-t Cut Problem

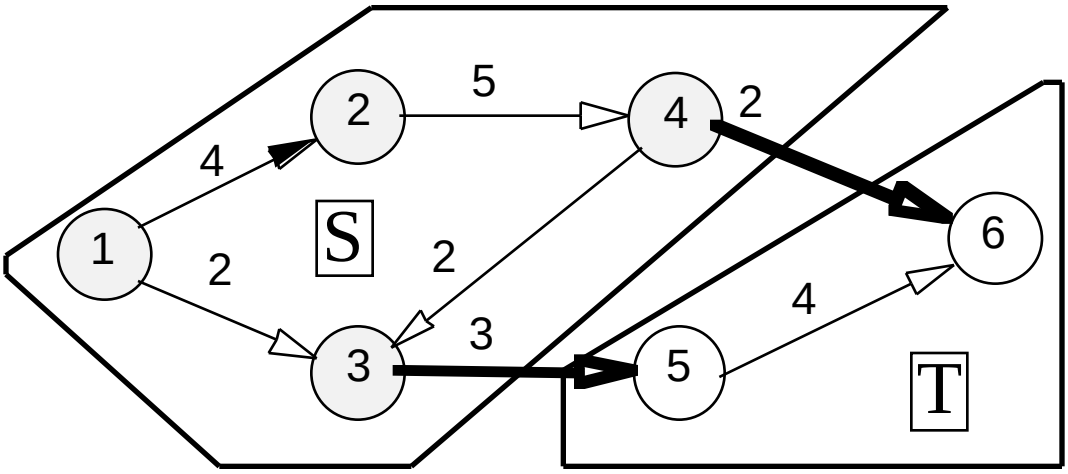
Let G = (N,A) denote a network, and let cij denote the capacity of arc (i,j) A.

A *cut* is a collection of arcs C such that there is no path from s to t in G' = (N, A/C)

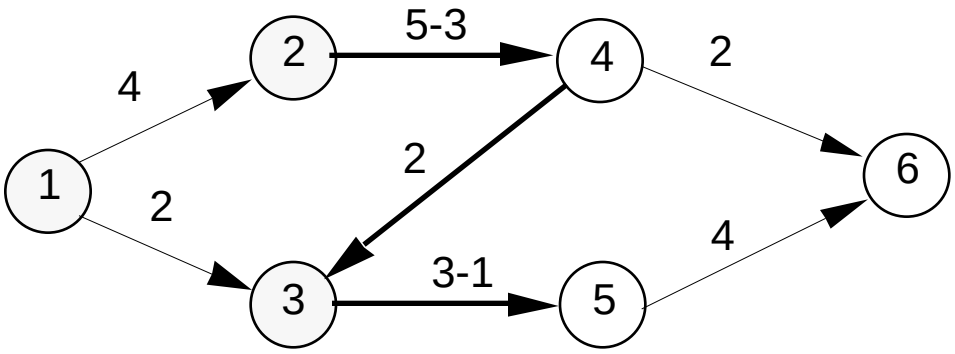
Example: Here is the proposed cut



Here is the minimum cut.



Here is the solution for the inverse problem



Original Problem

Minimize $\sum_{i,j} c_{ij} x_{ij}$

subject to $\sum_{(i,j)\ p} x_{ij} \geq 1$ for each p P[s,t]

$0 \leq x_{ij} \leq 1$

Claim. $\sum_{(i,j)\ p} x^*_{ij} = 1$ if p P[s,t] does not contain (4,3).

Let P'[s,t] = { p P[s,t] : p has no backward arc of x*}.

Inverse Problem

Minimize $\sum_{i,j} c_{ij} x_{ij}$

subject to $\sum_{(i,j)\ p} x_{ij} \geq 1$ for each p P'[s,t]

$0 \leq x_{ij} \leq 1$

Equivalently.

Let G' = (N, A'), where A' = A / backward arcs of x*

Then P'[s,t] is the set of paths from s to t in G'.

The inverse problem is equivalent to finding a minimum cut in G'.

Summary.

- If the original problem is a linear program, then the inverse problem is a linear program.

- If the original problem is solvable in polynomial time, then so is the inverse problem.

- If the original problem is a linear program, then the inverse problem can be defined as a linear program as restricted to the binding constraints plus some additional simple upper and lower bounds on the variables.

  If the original problem is a 0-1 linear program, and if x* is integral, then the inverse problem is a relaxation of the original problem.

- Simple solutions developed for:
  - inverse minimum cost spanning tree
  - inverse minimum cost flow
  - inverse matroid intersection
  - inverse minimum cut
  -