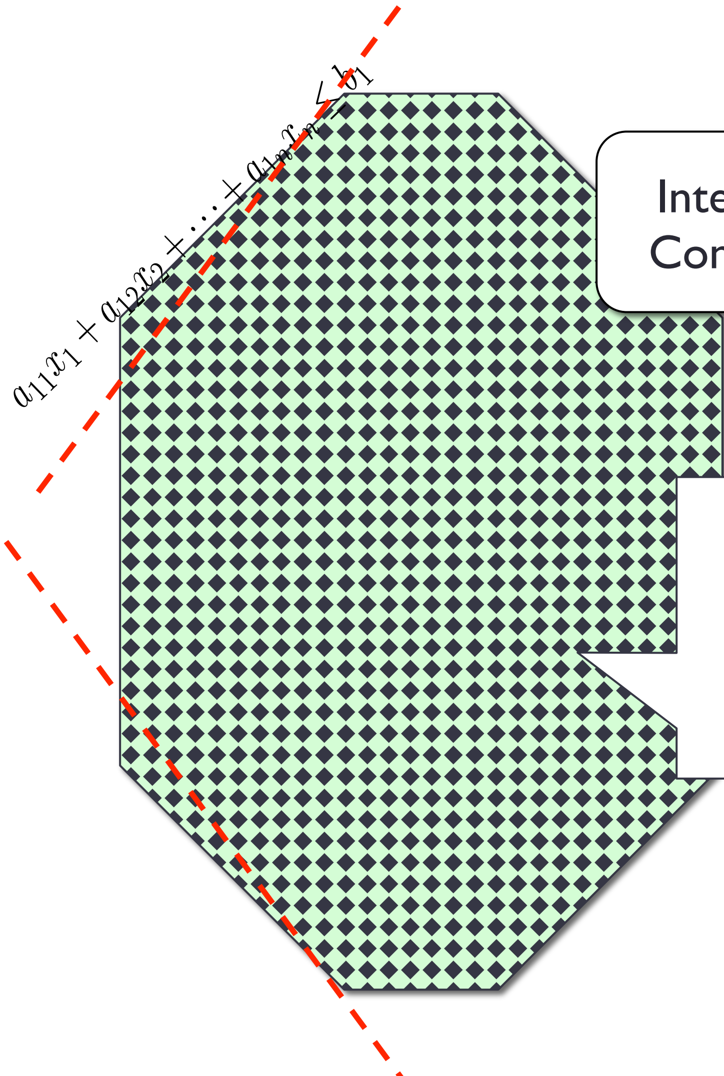
INTEGER LINEAR PROGRAMMING - INTRODUCTION

Integer Linear Programming



$$
\begin{array}{llllll}
\text{max} & c_1x_1 & +c_2x_2 & +\cdots+ & c_nx_n & \\
\text{s.t.} & a_{11}x_1 & +a_{12}x_2 & +\cdots+ & a_{1n}x_n & \leq \; b_1 \\
 & & & \ddots & & \vdots \\
 & a_{m1}x_1 & +a_{m2}x_2 & +\cdots+ & a_{mn}x_n & \leq \; b_m \\
 & \color{red}{x_1, \ldots, x_n} & \color{red}{\in \; \mathbb{Z}} & & &
\end{array}
$$

Integrality Constraint

Feasible Region: Z-Polyhedron
(n dimensional)

Integer Linear Programming

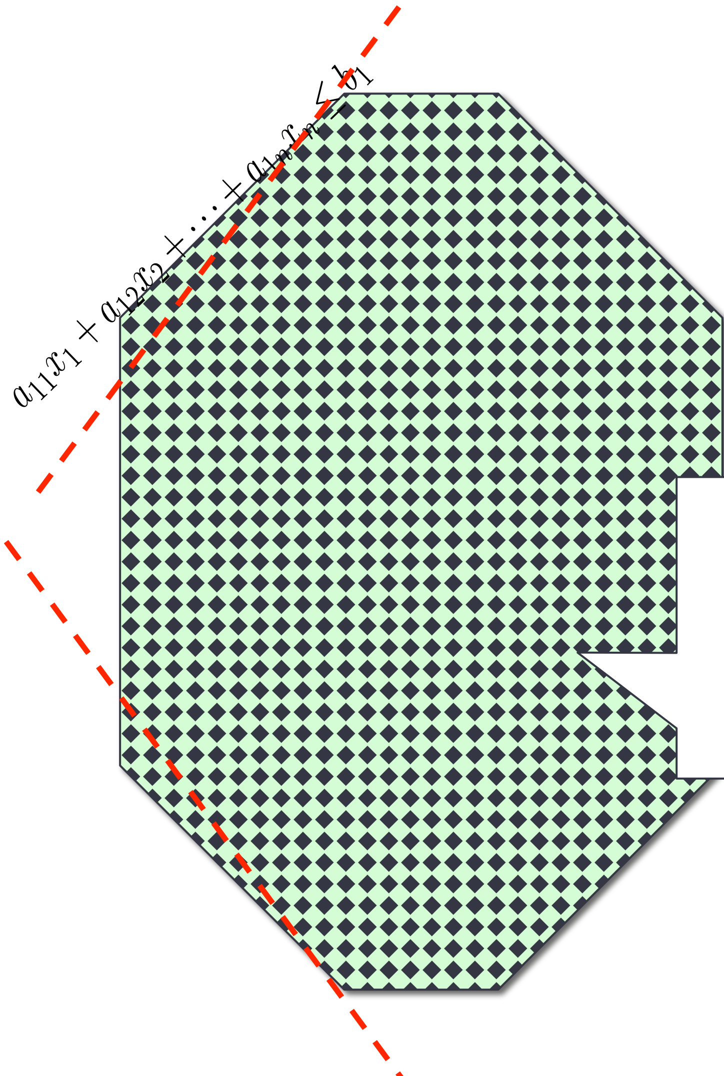
- Relaxation to a (real-valued) Linear Program
  - How does the LP relaxation answer relate to the ILP answer?
  - Integrality Gap
- Complexity of Integer Linear Programs
  - NP-Completeness
  - Some special cases of ILPs.
- Algorithms:
  - Branch-And-Bound
  - Gomory-Chvatal Cuts

INTEGER LINEAR PROGRAMMING: LP RELAXATION

1. Relax an ILP to an LP
2. Examples with same answers and different answers.
3. Integrality gap.

Integer Linear Programming

$$
\begin{array}{llllll}
\text{max} & c_1x_1 & +c_2x_2 & +\cdots+ & c_nx_n & \\
\text{s.t.} & a_{11}x_1 & +a_{12}x_2 & +\cdots+ & a_{1n}x_n & \leq & b_1 \\
 & & & \ddots & & \vdots & \\
 & a_{m1}x_1 & +a_{m2}x_2 & +\cdots+ & a_{mn}x_n & \leq & b_m \\
 & \multicolumn{5}{l}{\color{red}{x_1, \ldots, x_n \in \mathbb{Z}}}
\end{array}
$$



Feasible Region: Z-Polyhedron
(n dimensional)

Integer Linear Program

$$
\begin{array}{lccccc}
\text{max} & c_1x_1 & +c_2x_2 & +\cdots+ & c_nx_n & \\
\text{s.t.} & a_{11}x_1 & +a_{12}x_2 & +\cdots+ & a_{1n}x_n & \leq & b_1 \\
 & & & \ddots & & & \vdots \\
 & a_{m1}x_1 & +a_{m2}x_2 & +\cdots+ & a_{mn}x_n & \leq & b_m \\
 & & & & x_1, \ldots, x_n & \in & \mathbb{Z}
\end{array}
$$

- Feasibility of ILP:
  - *Integer feasible solution.*
- Unbounded ILP:
  - Integer feasible solutions can achieve arbitrarily large values for the objective.

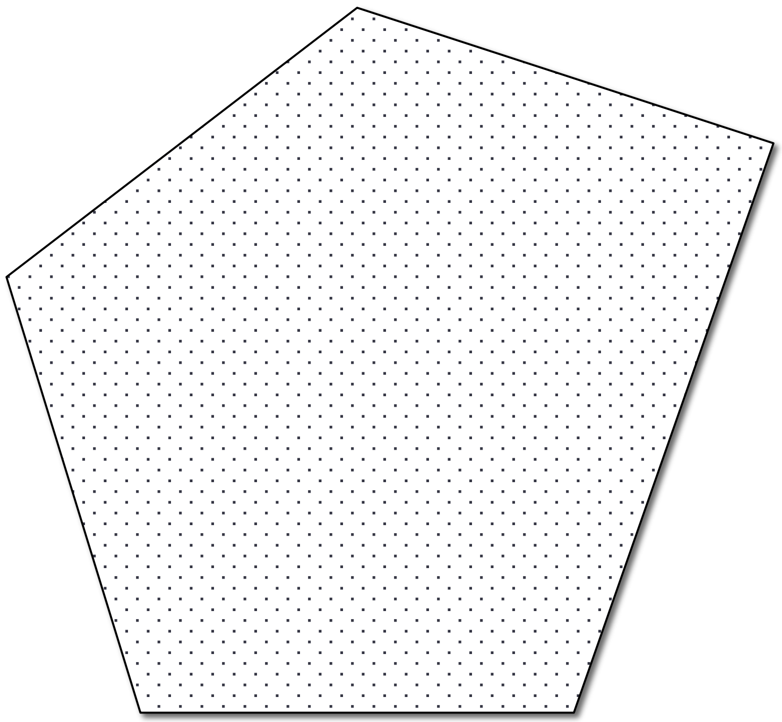
Linear Programming Relaxation

$$
\begin{array}{lcccccc}
\text{max} & c_1x_1 & +c_2x_2 & +\cdots+ & c_nx_n & & \\
\text{s.t.} & a_{11}x_1 & +a_{12}x_2 & +\cdots+ & a_{1n}x_n & \leq & b_1 \\
 & & & \ddots & & \vdots & \\
 & a_{m1}x_1 & +a_{m2}x_2 & +\cdots+ & a_{mn}x_n & \leq & b_m \\
 & x_1, \ldots, x_n & \in & \mathbb{Z} & & &
\end{array}
$$

Q: What happens to the answer if we take away the integrality constraints?

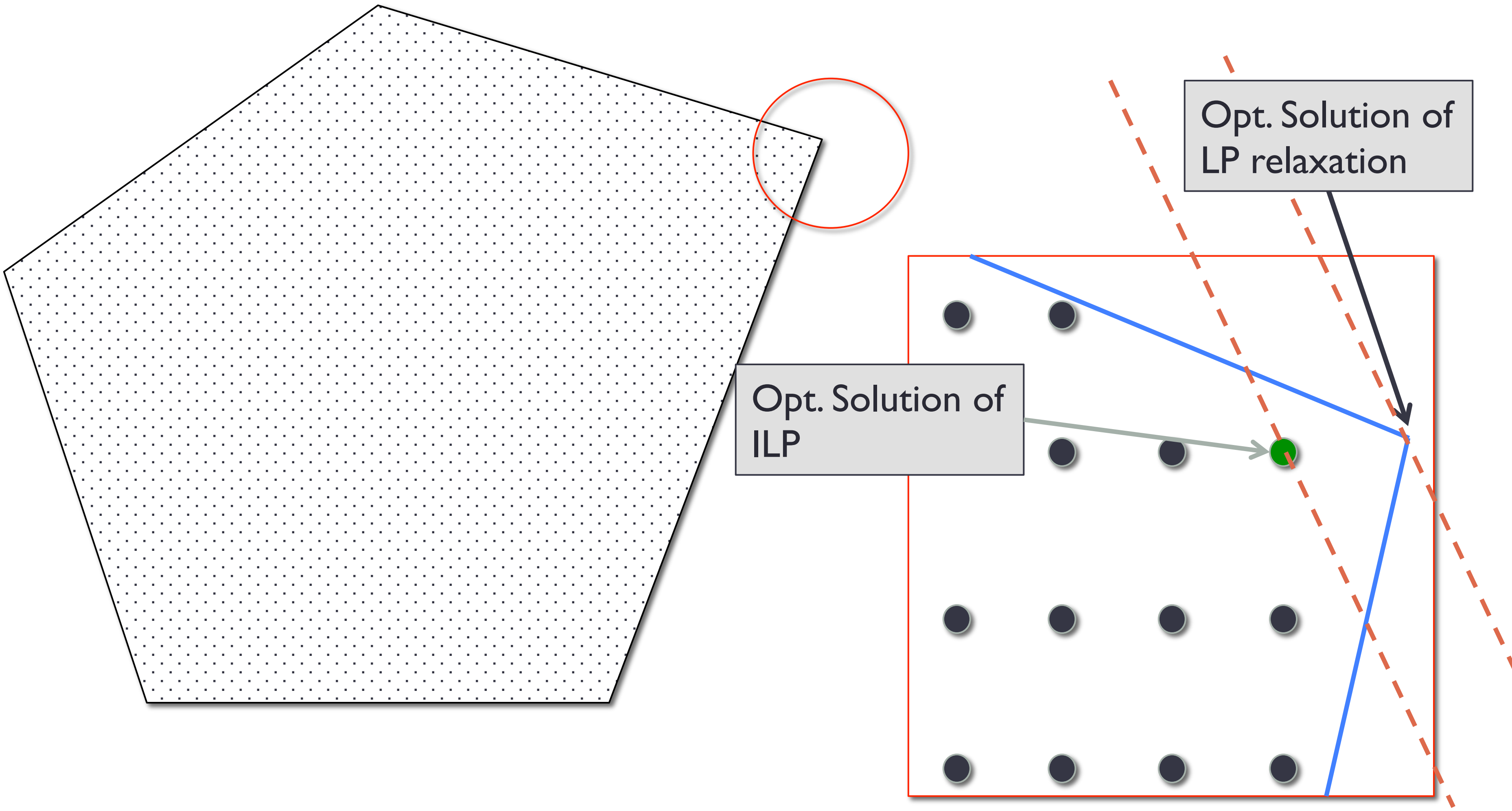
Feasible Regions

$$
\begin{array}{llllll}
\text{max} & c_1x_1 & +c_2x_2 & +\cdots+ & c_nx_n & & \\
\text{s.t.} & a_{11}x_1 & +a_{12}x_2 & +\cdots+ & a_{1n}x_n & \leq & b_1 \\
 & & & \ddots & & \vdots & \\
 & a_{m1}x_1 & +a_{m2}x_2 & +\cdots+ & a_{mn}x_n & \leq & b_m \\
 & & & x_1,\ldots,x_n & \in & \mathbb{Z} &
\end{array}
$$



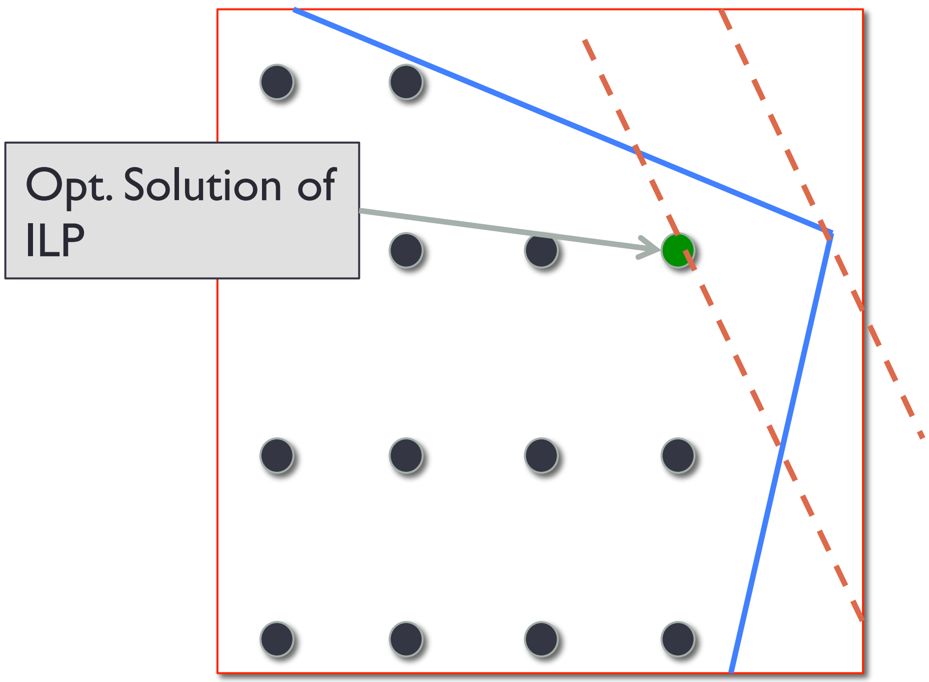
ILP feasible region $\subseteq$ LP feasible region

Case-1: Both LP and ILP are feasible.



Case-I

Optimal Objective of ILP ≤ Optimal solution of LP relaxation.



Example- I

Write down an example where LP optimum = ILP optimum

Example-2

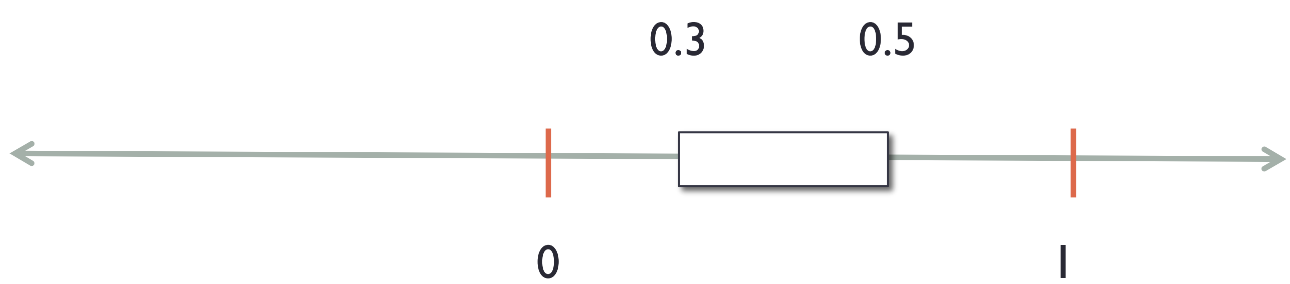
Write down an example where the two optima differ

Case-II: LP relaxation is feasible, ILP is infeasible.

$$\begin{aligned} \max \quad & x \\ \text{s.t.} \quad & \\ 3 \leq \quad & 10x \quad \leq 5 \end{aligned}$$

ILP is infeasible.

LP relaxation has optimal solution: 0.5

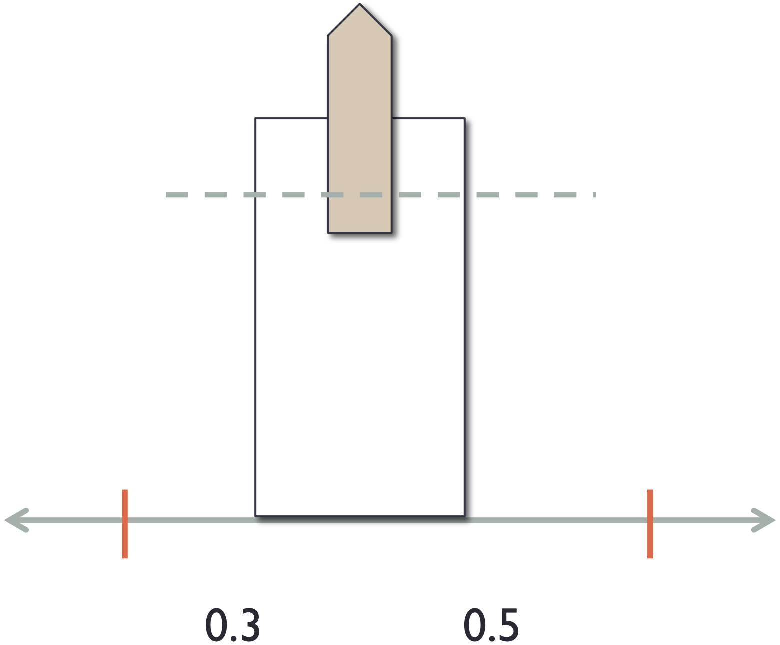


Case III: ILP is infeasible, LP is unbounded.

Example:

$$\begin{aligned} \max \quad & y \\ 3 \leq \quad & 10x \quad \leq 5 \\ 0 \leq \quad & y \end{aligned}$$

ILP is infeasible.
LP relaxation is unbounded



ILP outcomes vs. LP relaxation outcomes

Integer Linear Program (ILP)

LP Relaxation

	Infeasible	Unbounded	Optimal
Infeasible	Possible	Impossible	Impossible
Unbounded	Possible	Possible	Possible (*)
Optimal	Possible	Impossible	Possible

(*) Impossible if ILP has rational coefficients

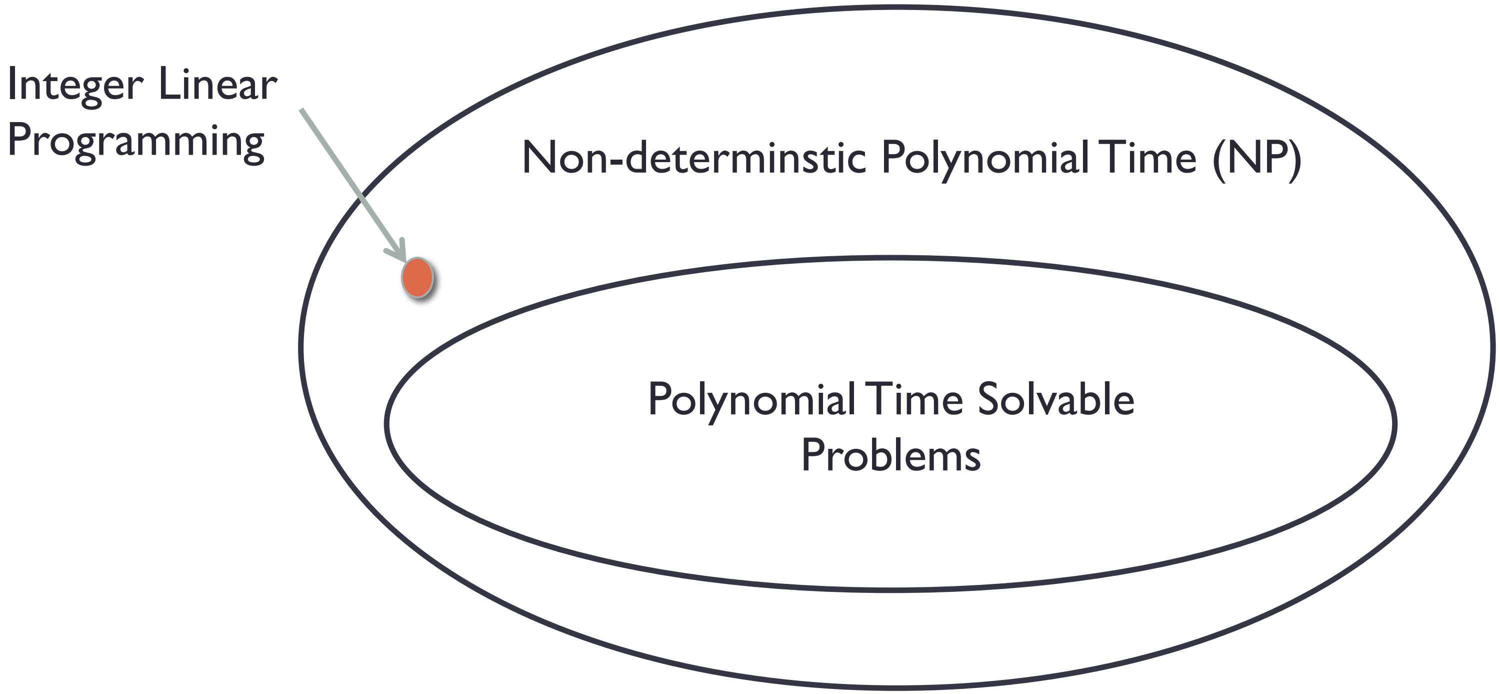
Summary (LP relaxation)

- LP relaxation: ILP minus the integrality constraints.
- LP relaxation's feasible region is a super-set of ILP feasible region.
- Analysis of various outcomes for ILP vs. outcomes for LP relaxations.

COMPLEXITY OF ILP

Complexity of Integer Linear Programs

Integer Linear Programming problems are NP-complete



Implications of P vs NP question

- P=NP
  - Considered an unlikely possibility by experts.
  - In this case, we will be able to solve ILPs in polynomial time.

- P != NP
  - In this case, we can show a non-polynomial lower bound on the complexity of solving ILPs.

Current State-of-the-art

- We have some very good algorithms for solving ILPs
  - They perform well on some important instances.
  - But, they all have exponential worst-case complexity.

- Compared to LPs,
  - The largest ILPs that we can solve are a 1000-fold smaller.

- Two strategies:
  - Try to solve the ILP
  - Find approximate answers for some special ILP instances.

ILP AND COMBINATORIAL OPTIMIZATION

Reducing 3-SAT to ILP

3-SAT Problem

x_1, x_2, x_3, x_4 Boolean Variables

$$(x_1 \text{ OR } x_2 \text{ OR } \neg x_3)$$
$$(\neg x_2 \text{ OR } \neg x_4 \text{ OR } x_1)$$
$$(x_1 \text{ OR } x_2 \text{ OR } \neg x_3)$$

Find values for Boolean variables

such that

All the Clauses are True.

3-SAT Problem (Infeasible/Unsat)

x_1, x_2, x_3, x_4 Boolean Variables

$$
\begin{array}{r}
(x_1 \text{ OR } \neg x_4 \text{ OR } x_2) \\
(\neg x_1 \text{ OR } \neg x_4 \text{ OR } x_2) \\
(x_4 \text{ OR } x_2) \\
(\neg\, x_2)
\end{array}
$$

No Boolean valuation satisfies all 4 clauses.

Reducing 3-SAT to ILP

$x_1, \ldots, x_n$ are Boolean variables.

$$\begin{array}{ll} C_1 : & (\ell_{1,1} \text{ OR } \ell_{1,2} \text{ OR } \ell_{1,3}) \\ \vdots & \ddots \\ C_m : & (\ell_{m,1} \text{ OR } \ell_{m,2} \text{ OR } \ell_{m,3}) \end{array}$$

m Clauses.

$\ell_{i,j}$ stands for a variable x_k or its negation $\neg x_k$

ILP reduction.

$$x_j \;\to\; y_j \in \{0, 1\}$$

False = 0
True = 1

$$\neg x_j \;\equiv\; (1 - y_j)$$

Clauses

Inequalities

$$(x_1 \text{ OR } x_2 \text{ OR } \neg x_5) \;\to\; y_1 + y_2 + (1 - y_5) \geq 1$$

Example-1

Convert this SAT problem to an ILP

$$(x_1 \text{ OR } x_2 \text{ OR } \neg x_3)$$

$$(\neg x_2 \text{ OR } \neg x_4 \text{ OR } x_1)$$

$$(x_1 \text{ OR } x_2 \text{ OR } \neg x_3)$$

Example-2

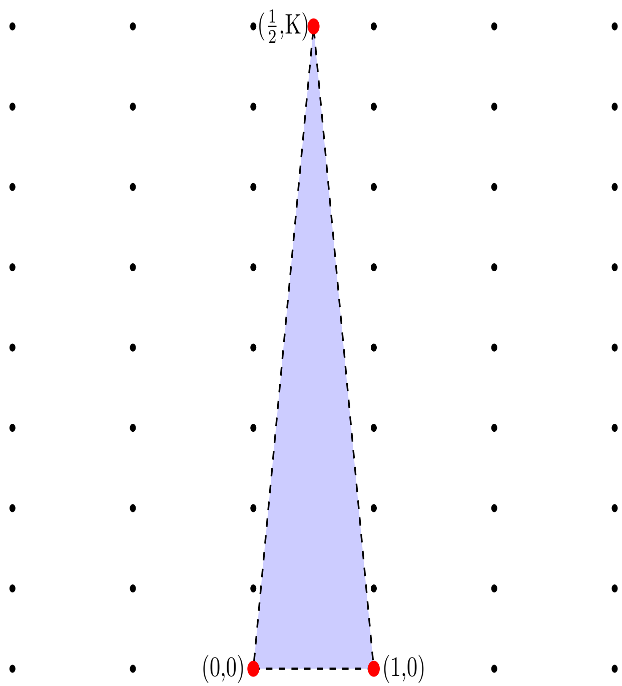
Convert this SAT problem to an ILP

$$(x_1 \text{ OR } \neg x_4 \text{ OR } x_2)$$
$$(\neg x_1 \text{ OR } \neg x_4 \text{ OR } x_2)$$
$$(x_4 \text{ OR } x_2)$$
$$(\neg\ x_2)$$

LP RELAXATION VS. ILP RELAXATION

Claim

LP relaxation's answer can be arbitrarily larger than the ILP's answer.



$$
\begin{array}{rrrcr}
\max & & x_2 & & \\
\text{s.t} & & x_2 & \geq & 0 \\
& 2Kx_1 & -x_2 & \geq & 0 \\
& -2Kx_1 & -x_2 & \geq & -2K \\
& x_1, & x_2 & \in & \mathbb{Z}
\end{array} .
$$

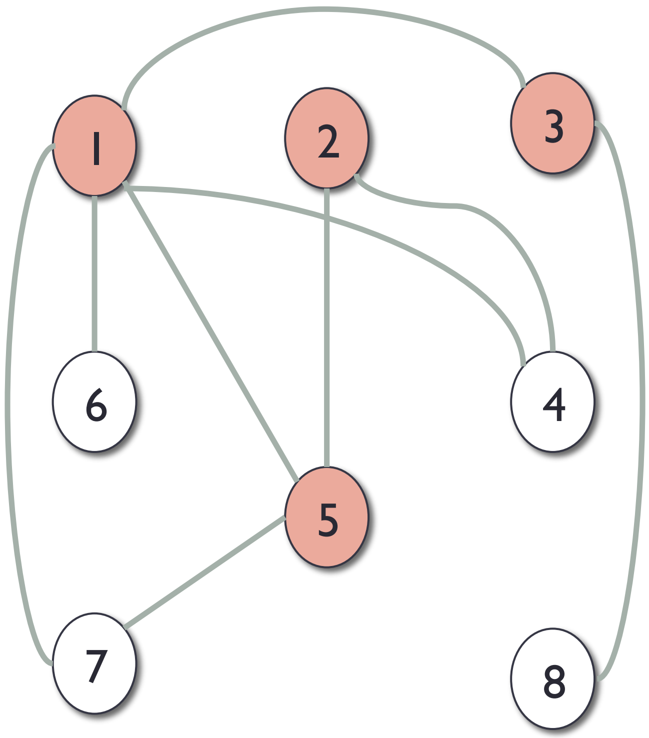
ILP AND VERTEX COVER

A flavor of approximation algorithms

Rounding Schemes

- LP relaxation yields solutions with fractional parts.
- However, ILP asks for integer solution.
- In some cases, we can approximate ILP optimum by "rounding"
  - Take optimal solution of LP relaxation
  - Round the answer to an integer answer using rounding scheme.
  - Deduce something about the ILP optimal solution.

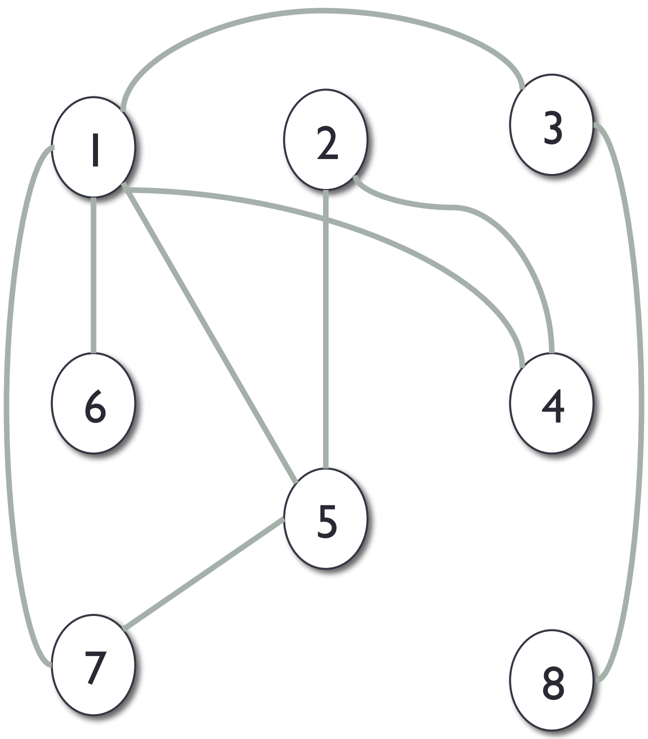
Vertex Cover Problem



Choose smallest subset of vertices
Every edge must be "covered"

Eg, { 1, 2, 3, 5 }
or
{1, 2, 3, 7 }

ILP for the vertex cover problem (Example)

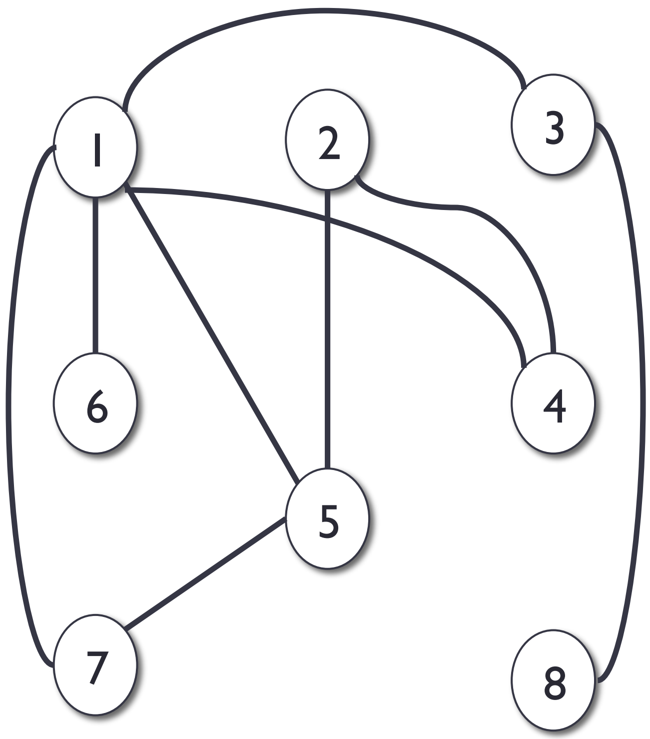


ILP decision variables

$$x_1, \ldots, x_8$$

$$x_i = \begin{cases} 0 & \text{Vertex } \# \ i \text{ not chosen in subset} \\ 1 & \text{Vertex } \# \ i \text{ is chosen in subset} \end{cases}$$

ILP for the vertex cover problem (Example)



$$
\begin{array}{llrcll}
\min & x_1 + x_2 + \cdots + x_8 & & & & \\
\text{s.t.} & & x_1 + x_7 & \geq & 1 & \leftarrow \text{Edge: } (1,7) \\
& & x_1 + x_6 & \geq & 1 & \leftarrow \text{Edge: } (1,6) \\
& & x_2 + x_4 & \geq & 1 & \\
& & \cdots & & & \\
& & x_i + x_j & \geq & 1 & \leftarrow (i,j) \in E \\
& & \cdots & & & \\
& & x_1 & \leq & 1 & \\
& & \vdots & & & \\
& & x_8 & \leq & 1 & \\
& & x_1, \ldots, x_8 & \geq & 0 & \\
& & x_1, \ldots, x_8 & \in & \mathbb{Z} &
\end{array}
$$

Vertex Cover to ILP

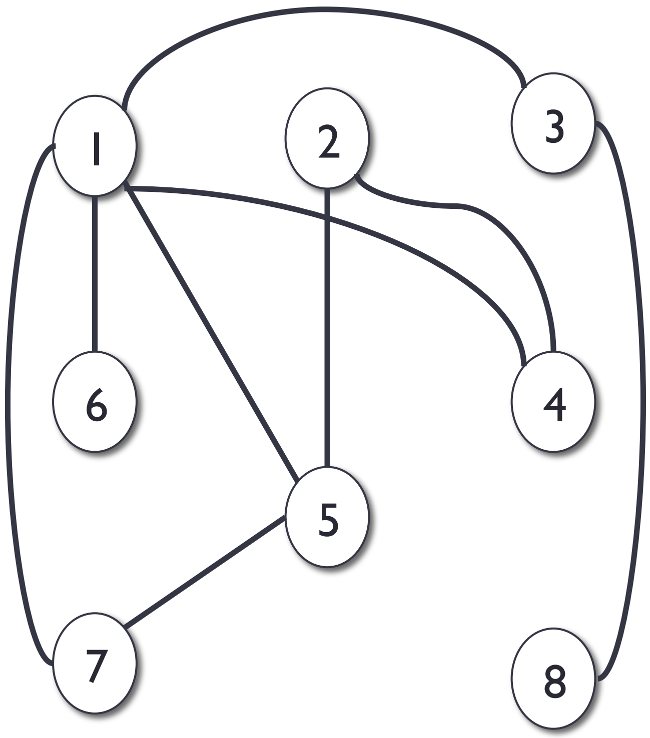
- Vertices {1,…, n}
  - Decision variables: $x_1, \ldots, x_n$

$$\boxed{x_i \in \{0,1\}}$$

$$\begin{array}{lll}
\min & \sum_{i=1}^{n} x_i & \\
\text{s.t.} & 0 \leq x_i \leq 1 & \forall\, i \in V \\
& x_i + x_j \geq 1 & \forall\, (i,j) \in E \\
& x_i \in \mathbb{Z} & \forall\, i \in V
\end{array}$$

LP relaxation of a vertex cover

- Problem: we may get fractional solution.



x_1	1
x_2	1
x_3	$\frac{3}{4}$
x_4	0
x_5	$\frac{5}{6}$
x_6	0
x_7	$\frac{1}{6}$
x_8	$\frac{1}{4}$

Objective value: 4

But solution meaningless for vertex cover.

Rounding Scheme

- Simple rounding scheme:

$$x_i^* \geq \frac{1}{2} \quad \rightarrow \quad x_i = 1$$

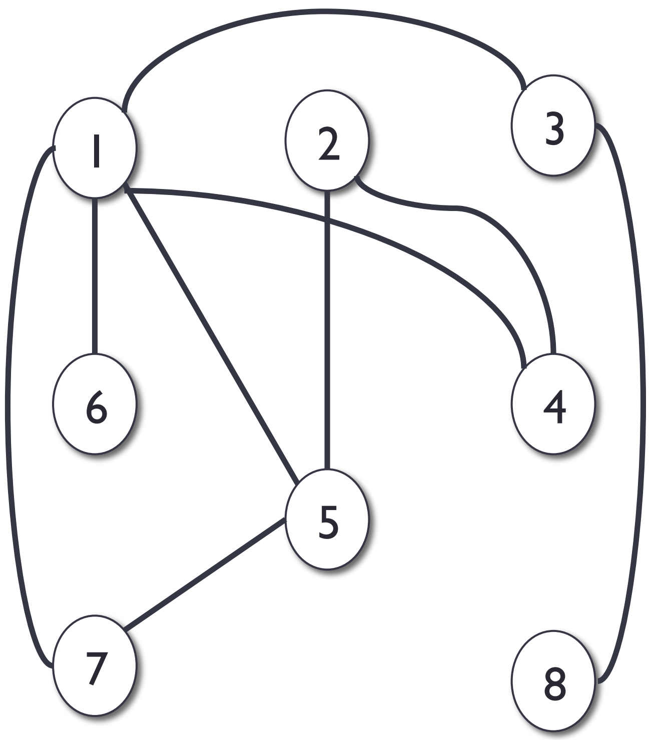
Real-Optimal Solution is at least 0.5

Include vertex in the cover.

$$x_i^* < \frac{1}{2} \quad \rightarrow \quad x_i = 0$$

LP relaxation of a vertex cover

- Problem: we may get fractional solution.



x_1	1
x_2	1
x_3	$\frac{3}{4}$
x_4	0
x_5	$\frac{5}{6}$
x_6	0
x_7	$\frac{1}{6}$
x_8	$\frac{1}{4}$

x_1	1
x_2	1
x_3	1
x_4	0
x_5	1
x_6	0
x_7	0
x_8	0

Rounding Scheme

Rounding scheme takes optimal fractional solution from LP relaxation and produces an integral solution.

$$\mathbf{x}^* \quad \xrightarrow{\text{rounding}} \quad \hat{\mathbf{x}}$$

1. Does rounding always produces a valid vertex cover?
2. How does the rounded solution compare to the opt. solution?

Rounding Scheme Produces a Cover

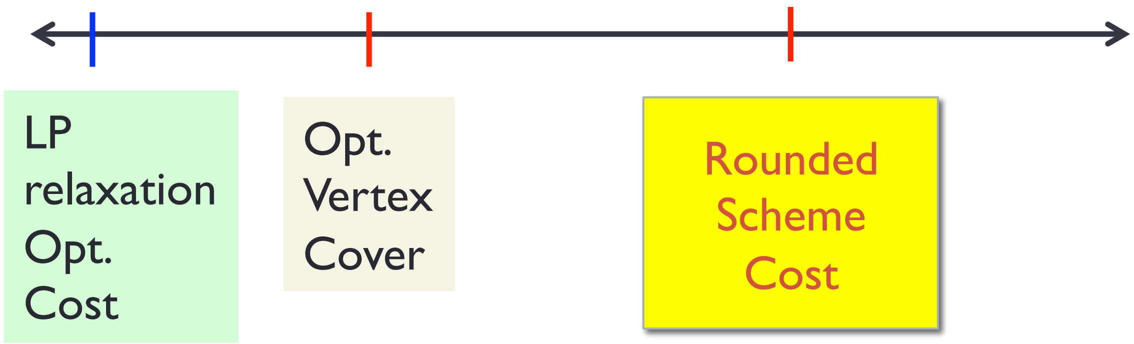
$$\mathbf{x}^* \xrightarrow{\text{rounding}} \hat{\mathbf{x}}$$

$$x_i^* + x_j^* \geq 1, \quad \text{for each } (i,j) \in E$$

$$\hat{x}_i = 1 \text{ or } \hat{x}_j = 1 \text{ for each } (i,j) \in E$$

To Prove: The solution obtained after rounding covers every edge.

Rounding Scheme Approximation Guarantee



$$\mathbf{x}^* \xrightarrow{\text{rounding}} \hat{\mathbf{x}}$$

Fact: $2x_i^* \geq \hat{x}_i$ for all vertices i.

$$2\sum_{i=1}^{n} x^* \geq \sum_{i=1}^{n} \hat{x}_i$$

$2 * (\text{Cost of LP relaxation}) \geq (\text{Cost of Rounded Scheme Vertex Cover})$

Approximation Guarantee

- Theorem #1: Rounding scheme yields a vertex cover.
- Cost of the solution obtained by rounding: C
- Optimal vertex cover cost: C^*

- Theorem #2: $C^* \leq C \leq 2\,C^*$

- LP relaxation + rounding scheme:
  - 2-approximation for vertex cover!!

SOLVING ILP USING GLPK

Specifying integer variables in Mathprog

GLPK integer solver

- GLPK has a very good integer solver.
  - Uses branch-and-bound + Gomory cut techniques
  - We will examine these techniques soon.

- In this lecture,
  - Show how to solve (mixed) integer linear programs
  - Continue to use AMPL format.

- This is the best option for solving ILPs/MIPs

Example-I (ILP)

$$
\begin{array}{rcccccccc}
\min & x_1 & +x_2 & +x_3 & +x_4 & +x_5 & +x_6 & & \\
 & x_1 & +x_2 & & & & & \geq & 1 \\
 & x_1 & +x_2 & & & & +x_6 & \geq & 1 \\
 & & & x_3 & +x_4 & & & \geq & 1 \\
 & & & x_3 & +x_4 & +x_5 & & \geq & 1 \\
 & & & & x_4 & +x_5 & +x_6 & \geq & 1 \\
 & & x_2 & & & +x_5 & +x_6 & \geq & 1 \\
 & x_1, & x_2, & x_3, & x_4, & x_5, & x_6 & \in & \mathbb{Z}
\end{array}
$$

Specifying variable type

var x; # specifies a real-valued decision variable

var y integer; # specifies an integer variable

var z binary; # specifies a binary variable

Example – I expressing in AMPL

```
var x{1..6} integer;
# Declare 6 integer variables
minimize obj: sum{i in 1..6} x[i];
c1: x[1] + x[2] >= 1;
c2: x[1] + x[2] + x[6] >= 1;
c4: x[3] + x[4]  >= 1;
c5: x[3]+ x[4] + x[5] >= 1;
c6: x[4] + x[5] + x[6]   >= 1;
c7: x[2] + x[5] + x[6] >= 1;
solve;
display{i in 1..6} x[i];
end
```

$$
\begin{array}{lccccccll}
\min & x_1 & +x_2 & +x_3 & +x_4 & +x_5 & +x_6 & & \\
 & x_1 & +x_2 & & & & & \geq & 1 \\
 & x_1 & +x_2 & & & & +x_6 & \geq & 1 \\
 & & & x_3 & +x_4 & & & \geq & 1 \\
 & & & x_3 & +x_4 & +x_5 & & \geq & 1 \\
 & & & & x_4 & +x_5 & +x_6 & \geq & 1 \\
 & & x_2 & & & +x_5 & +x_6 & \geq & 1 \\
 & x_1, & x_2, & x_3, & x_4, & x_5, & x_6 & \in & \mathbb{Z}
\end{array}
$$

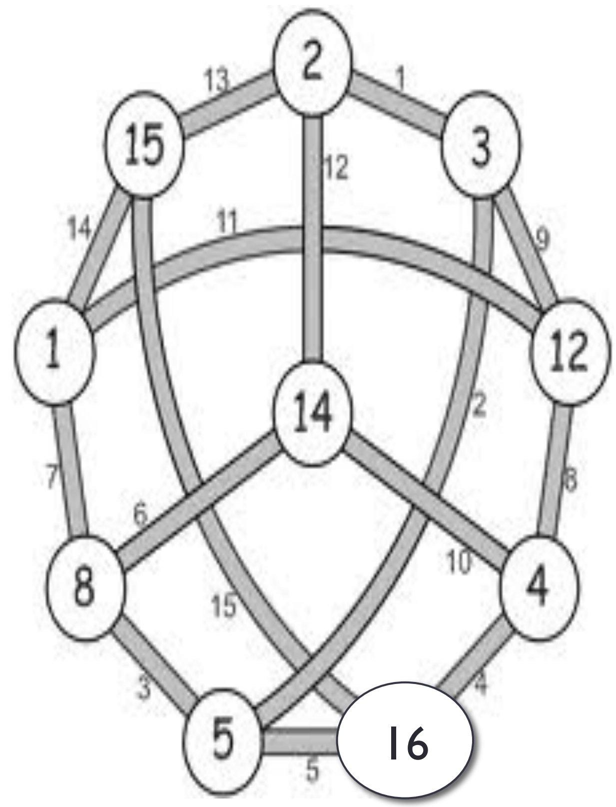
Example-1 Solving using GLPK

- glpsol -- math ip1.math

```
Display statement at line 25
x[1].val = 0
x[2].val = 1
x[3].val = 0
x[4].val = 1
x[5].val = 0
x[6].val = 0
Model has been successfully processed
```

Example -2

Vertex Cover Problem



source mathpuzzle.com

Vertex Cover to ILP

- Vertices {1,…, n}
  - Decision variables: $x_1, \ldots, x_n$

$$x_i \in \{0, 1\}$$

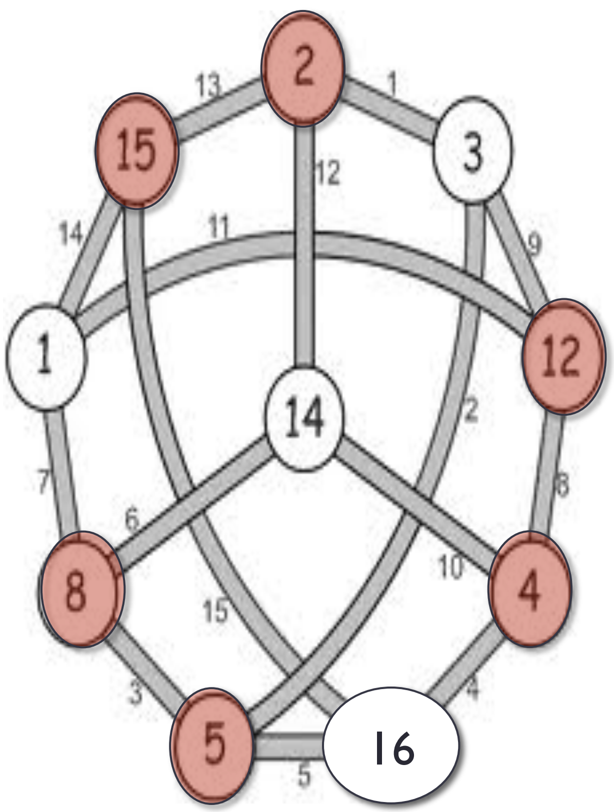
$$\begin{array}{llr}
\min & \sum_{i=1}^{n} x_i & \\
\text{s.t.} & 0 \leq x_i \leq 1 & \forall\, i \in V \\
& x_i + x_j \geq 1 & \forall\, (i, j) \in E \\
& x_i \in \mathbb{Z} & \forall\, i \in V
\end{array}$$

Vertex Cover AMPL (Model + Data)

```
param n;
var x {1..n} binary;
# binary specifies that the variables are binary

set E within {i in 1..n, j in 1..n: i < j};
# specify that the edges will be a set.
# each edge will be entered as (i,j) where i < j

minimize obj: sum{i in 1..n} x[i];
# minimize cost of the cover
s.t.
c{(i,j) in E}: x[i] + x[j] >= 1;

solve;
display{i in 1..n} x[i];
```

```
data;
param n := 16;

set E := (2,3) (3,5) (5,8)
         (4,16) (5,16) (8,14)
         (1,8) (4,12) (3,12) (4,14)
         (1,12) (2,14) (2,15) (1,15) (15,16) ;

end;
```

Running GLPK …



- glpsol -m vertexCover.model

x[1].val = 0
x[2].val = 1
x[3].val = 0
x[4].val = 1
x[5].val = 1
x[6].val = 0
x[7].val = 0
x[8].val = 1
x[9].val = 0
x[10].val = 0
x[11].val = 0
x[12].val = 1
x[13].val = 0
x[14].val = 0
x[15].val = 1
x[16].val = 0

SOLVING ILPS IN MATLAB/ OCTAVE

MATLAB Optimization Package

- Supports solving binary integer programming problem
- “bintprog function”
- Same interface as linprog.
  - Except that all variables are assumed binary.

- Uses branch-and-bound
  - Not considered to be a good implementation.

CVX

- Unfortunately, does not support integer programming in the free version.

- Links to commercial tools Gurobi/MOSEK/CPLEX
  - Powerful state of the art integer solvers.
  - They make it available to academic users for free.

- We will continue to use GLPK for MATLAB/Octave.

Solution for MATLAB

- We will use glpkmex: a glpk interface to matlab and octave.

http://sourceforge.net/projects/glpkmex/

- Octave users may already know about this interface.

- It implements a convenient function glpk(..)

Over to matlab demo…