15.053/8 April 9, 2013

Integer Programming 5

- Gomory cuts
- Constraint generation
- Formulations using set notations

Quote of the Day

“Mathematicians are like Frenchmen: whatever you say to them, they translate into their own language, and forthwith it is something entirely different.”

Johann Wolfgang von Goethe

Gomory Cuts

Gomory cuts is a general method for adding valid inequalities (also known as cuts) to all MIPs

- **Gomory cuts are VERY useful to improve bounds.**
- **Gomory cuts are obtained from a single constraint of the optimal tableau for the LP relaxation.**
- **Assume here that all variables must be integer valued.**

Case 1: All LHS coefficients are between 0 and 1.

$$.2\,x_1 + .3\,x_2 + .3\,x_3 + .5\,x_4 + x_5 = 1.8 \qquad (1)$$

Valid inequality (ignore contribution from x_5) :

$$.2\,x_1 + .3\,x_2 + .3\,x_3 + .5\,x_4 \geq .8 \qquad (2)$$

Case 2: LHS coefficients are ≥ 0.

Case 2: all LHS coefficients are non-negative

$$1.2\,x_1 + .3\,x_2 + 2.3\,x_3 + 2.5\,x_4 + x_5 = 4.8 \qquad (1)$$

Valid inequality (focus on fractional parts):

$$.2\,x_1 + .3\,x_2 + .3\,x_3 + .5\,x_4 \geq .8 \qquad (2)$$

The fractional part of

$$\text{“}1.2\,x_1 + .3\,x_2 + 2.3\,x_3 + 2.5\,x_4 + x_5\text{”}$$

is the same as that of

$$\text{“}.2\,x_1 + .3\,x_2 + .3\,x_3 + .5\,x_4\text{”}$$

Gomory cuts: general case

Case 3: General case

$$1.2\,x_1 - 1.3\,x_2 - 2.4\,x_3 + 11.8\,x_4 + x_5 = 2.9 \qquad (1)$$

Round down (be careful about negatives):

$$1\,x_1 - 2\,x_2 - 3\,x_3 + 11\,x_4 + x_5 \le 2 \qquad (2)$$

Valid inequality: subtract (2) from (1):

$$.2\,x_1 + .7x_2 + .6\,x_3 + .8\,x_4 \ge .9 \qquad (3)$$

The coefficients of the valid inequality are:

- fractional parts of (1)
- non-negative

Another Gomory Cut

$$x_1 + -2.9\,x_2 + -3.4\,x_3 + 2.7\,x_4 = 2.7 \qquad (1)$$

Round down

$$x_1 + -3\,x_2 + -4\,x_3 + 2\,x_4 \leq 2 \qquad (2)$$

Then subtract (2) from (1) to get the Gomory cut

$$.1\,x_2 + .6\,x_3 + .7\,x_4 \geq .7 \qquad (3)$$

Note: negative coefficients also get rounded down.

x_1	x_2	x_3	x_4	x_5		
1.6	- 4.7	3.2	-1.4	1	=	9.4

What is the Gomory Cut?

1. $x_1 - 4x_2 + 3x_3 - x_4 + x_5 \leq 9$
2. $x_1 - 5x_2 + 3x_3 - 2x_4 + x_5 \leq 9$
3. $.6x_1 - .7x_2 + .2x_3 - .4x_4 \geq .4$
4. $.6x_1 + .3x_2 + .2x_3 + .6x_4 \geq .4$
5. none of the above

Does a Gomory cut always exist? Yes!

x_1	x_2	x_3	x_4	x_5		
1.6	- 4.7	3.2	-1.4	1	=	9.4

If the RHS in the final tableau is integer, then the bfs is integer, and we have solved the LP.

Otherwise, there is a non-integer in the RHS.

If all coefficients on the LHS of this constraint are integer, then there is no way of satisfying the constraint.

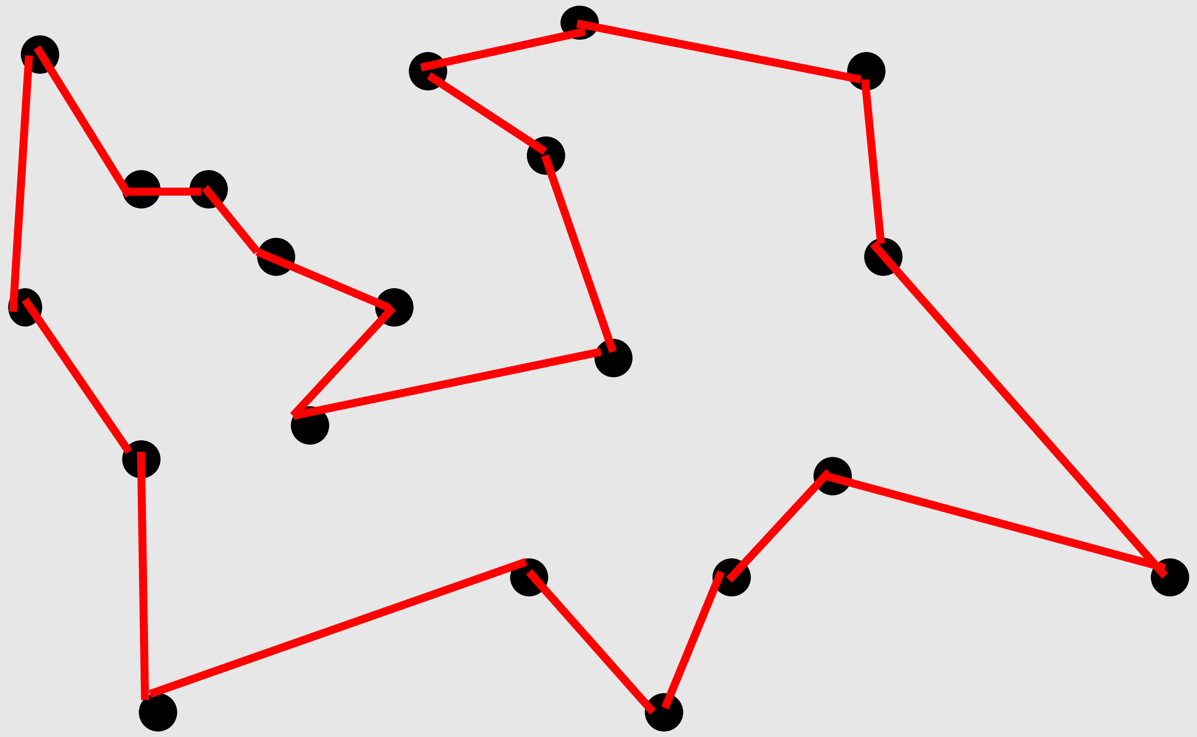
Therefore, there are 1 or more fractional coefficients.

All of these are for non-basic variables. These are used for the Gomory cut.

A brief history of Gomory cuts

- Gomory -- 1963
- Initial use : solving IPs without Branch and Bound
  - too slow for this type of use
- Conventional wisdom prior to 1990: Gomory cuts are not useful in practice
- Balas, Ceria, Cornuejols tested Gomory cuts in the early 1990s with great success.
- Current conventional wisdom: Gomory cuts are extremely useful for solving IPs in practice.

Traveling Salesman Problem (TSP)



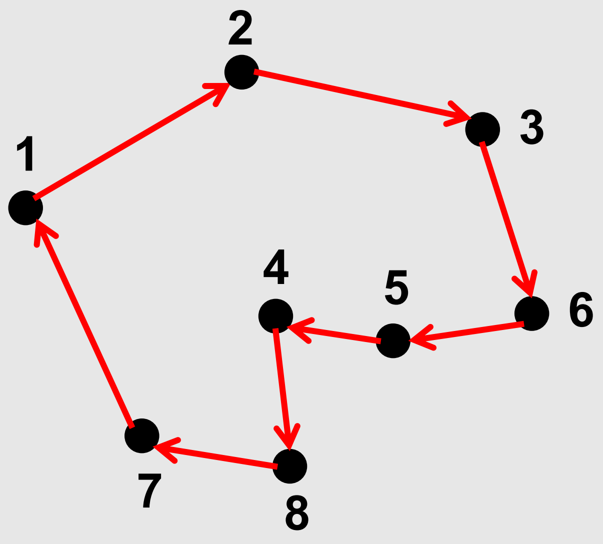
What is a minimum length tour that visits each point?

Comments on the TSP

- Very well studied problem
- Applications: vehicle routing, laser drilling in integrated circuits, manufacturing, and more.
- Large instances have been solved optimally (5000 cities and larger)
- Very large instances have been solved approximately (10 million cities to within a couple of percent of optimum.)
- We will formulate it by adding constraints that look like cuts

An Integer Programming Formulation

$$x_{ij} = \begin{cases} 1 & \text{if city } j \text{ follows city } i \text{ in the tour} \\ 0 & \text{otherwise} \end{cases}$$



c_{ij} = distance from city i to city j

Minimize $\boxed{\text{objective?}}$

$\boxed{\text{Constraint?}}$ for each j = 1 to n

$\boxed{\text{Constraint?}}$ for each i = 1 to n

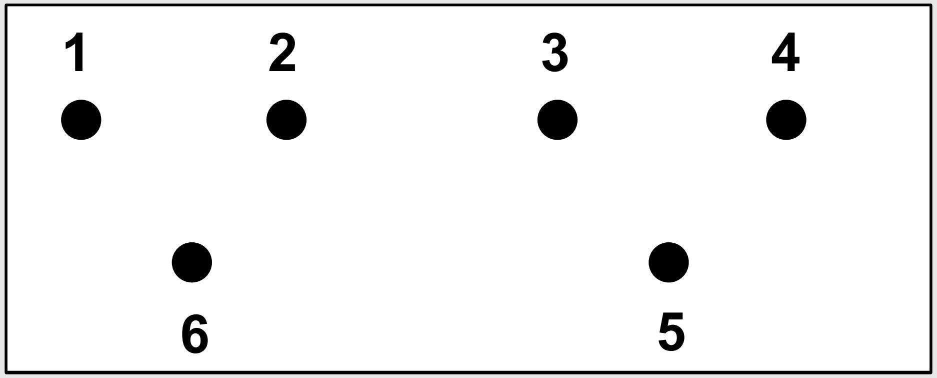
The salesperson enters city j for each j.

The salesperson leaves city i for each i.

$x_{ij} \in \{0,1\}$ for each i, j = 1 to n

Is this enough?

The previous IP needs more constraints



What is the optimal solution satisfying the IP constraints?

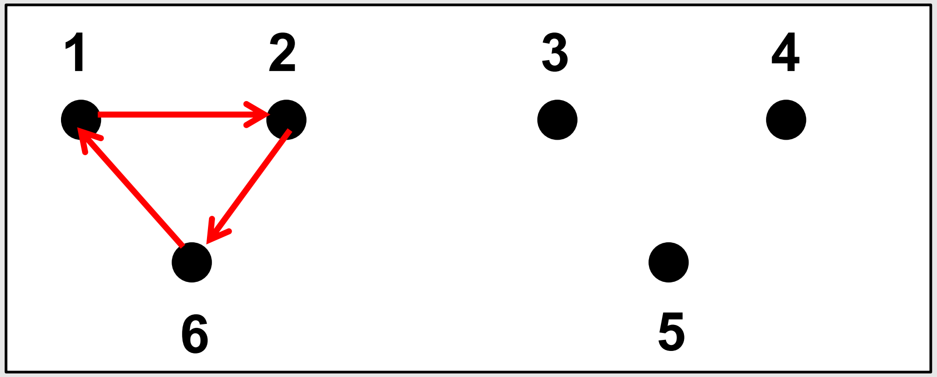
The salesperson leaves each city.

The salesperson enters each city.

But the tour is not connected.

It consists of two subtours.

Subtour elimination constraints



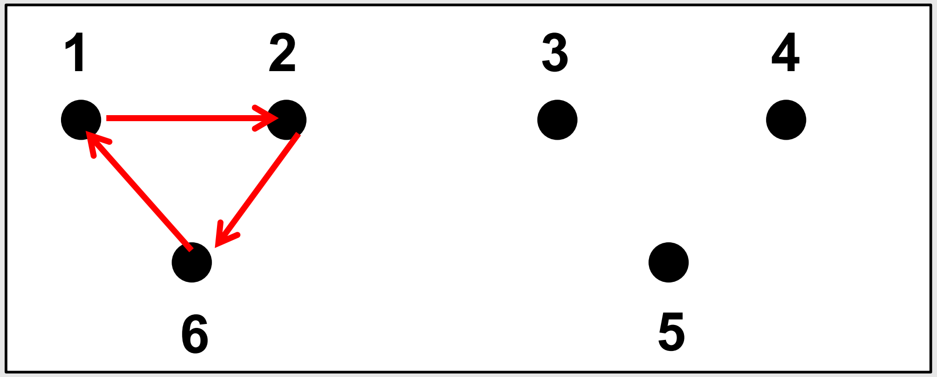
We want to add constraints so that no subtour is feasible.

To ensure that 1-2-6 is not a subtour:

- There is at least one edge of the tour from city 1, 2 or 6 to city 3, 4 or 5.

$$\sum_{\substack{i=1,2,\text{or } 6 \\ j=3,4,\text{or } 5}} x_{ij}$$

Subtour elimination constraints: general case



$$\sum_{\substack{i=1,2,or\,6\\ j=3,4,or\,5}} x_{ij} \geq 1$$

Let S ⊆ {1, 2, …, n}, the set of cities. Let N\S be the other cities.

For any proper subset S of cities, there is an edge that leaves S and enters N\S.

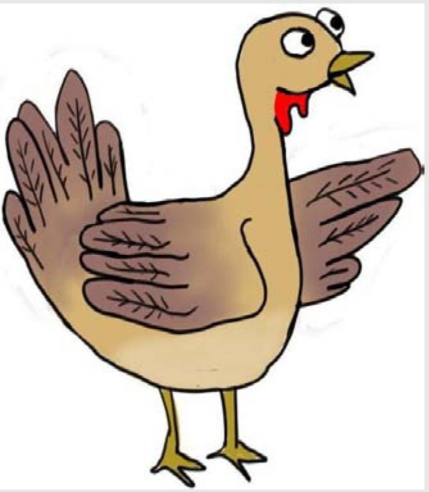
$$\sum_{i\in S,\, j\in N\setminus S} x_{ij} \geq 1$$

But if there are 100 variables that must be more than a billion constraints.

Are you trying to make my head hurt?

Actually, the number of constraints is around 2^{100}, which is more than a billion times a billion times a billion.

But, these constraints are still useful in practice.



The reason it is useful in practice is that we only include constraints as they are needed.

Constraint generation

- When an LP has too many constraints, e.g., an exponentially large number, constraint generation can be used. Let **C** denote the set of all constraints.
- The LP is solved with a subset $S \subseteq C$ of the constraints. An optimal solution x* is obtained to this LP.
- If x* violates a constraint in **C** that are not in **S**, then this constraint is added to **S**.
- This approach works if one can efficiently find a violated constraint in $C \backslash S$.

Constraint generation for the LP relaxation of the TSP

$$\text{Min} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \qquad (1)$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \forall j \qquad (2)$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \forall i \qquad (3)$$

$$0 \le x_{ij} \le 1 \text{ for all } i, j \qquad (4)$$

$$\sum_{i \in S, j \in N \setminus S} x_{ij} \ge 1 \qquad (5)$$

as needed

Solve the LP relaxation without (5).

If the solution violates some of the constraints of (5), add them to the LP and solve again.

Continue until we have no more constraints to add.

Often, this LP solution value is very close to the IP optimal solution value. (less than 2% error).

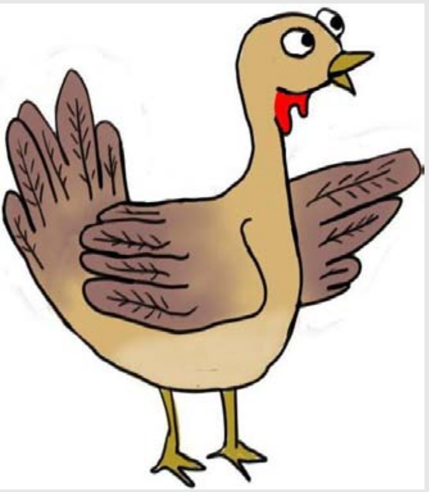
Are you saying that you can really solve an LP even though there are trillions of constraints?

My head hurts.

Yes! This technique really works, and it is used in practice.

There are related approaches in which one can solve LPs with trillions of variables.

They are called column generation approaches.



But we have done enough for today. If you want to learn more, you can read about column generation techniques in AMP in the chapter on decomposition.

Mental Break

Rest of the lecture

- Practice with additional applications of IPs
- A common feature of the modeling: there is a focus on subsets of a “ground set.”
- Then we will conclude the segment on integer programming

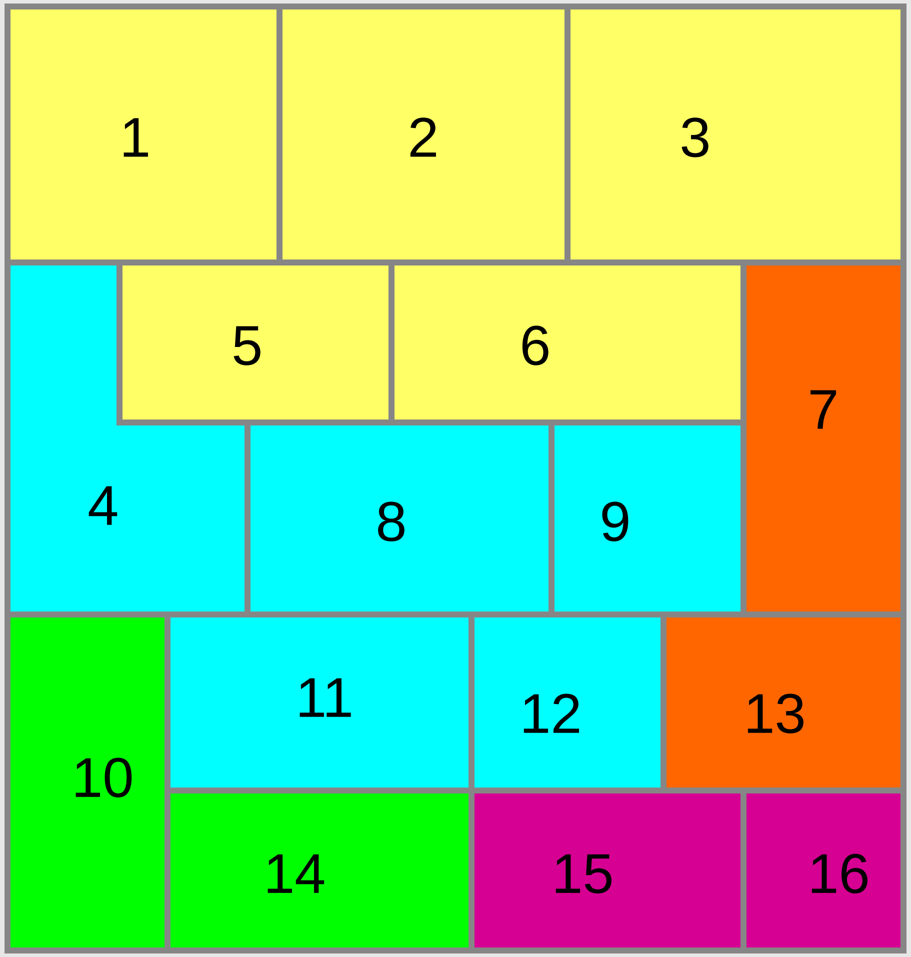
Subsets of a ground set

- In some problems, we start with a collection of items. This is called the ***ground set.***
  - e.g., set of items to choose from for the knapsack problem
- The ground set is sometimes denoted as {1, 2, 3, …, n}.
- Sometimes, we formulate constraints using notation that refers to **subsets of the ground set**; e.g., we may write the constraint
  - "$x_1 + x_3 + x_5 + x_9 \leq 1$" as

$$\sum_{j \in S} x_j \leq 1 \qquad \text{where } S = \{1, 3, 5, 9\}$$

053 Chocolates

Amit and Mita have started a new chocolate company.



Locate 053 Chocolate stores so that each district has a store in it or next to it.

Minimize the number of stores needed.

How do we represent this as an IP?

Variables

$x_j = 1$ if a chocolate store is put in district j

$x_j = 0$ otherwise

Objective

Minimize $x_1 + x_2 + \ldots + x_{16}$

Constraints

s.t. $x_1 + x_2 + x_4 + x_5 \geq 1$

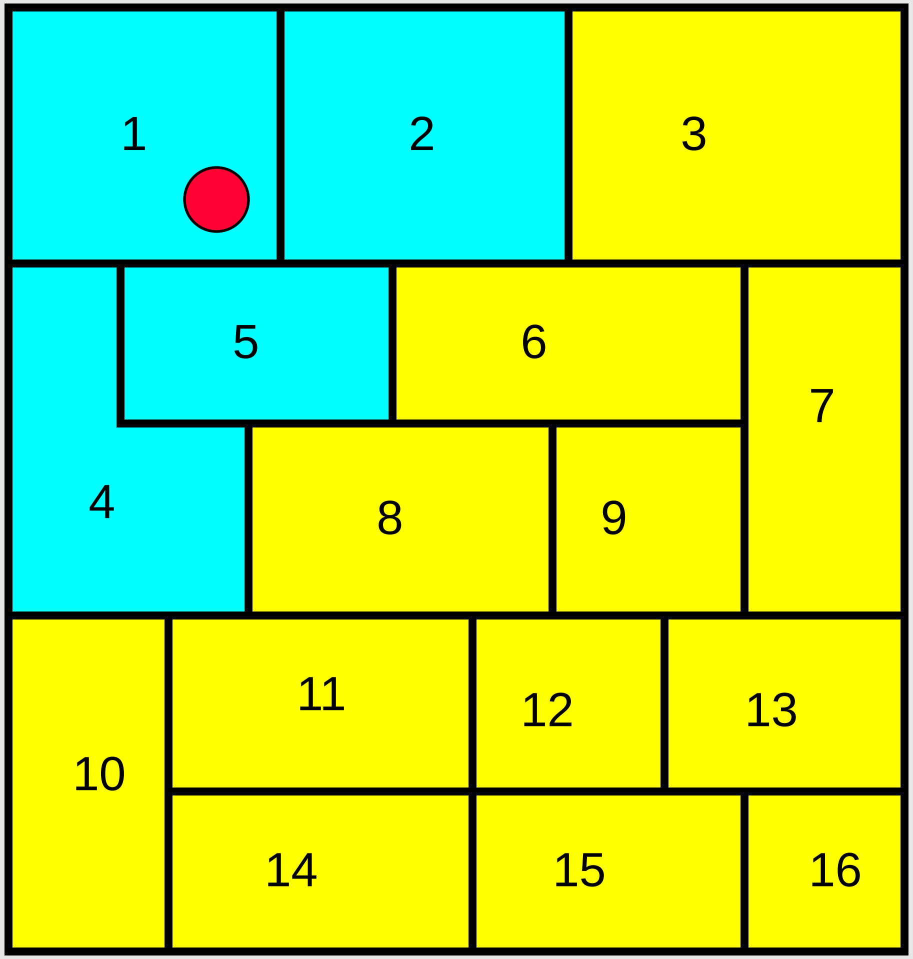
$x_1 + x_2 + x_3 + x_5 + x_6 \geq 1$

$\vdots \qquad \vdots$

$x_{13} + x_{15} + x_{16} \geq 1$

$x_j \in \{0, 1\}$ for each j.

Representation as Set Covering Problem



#	Subset S(j)
1	{1, 2, 4, 5}

S(j) is j plus all of the districts that share a border with j.

Putting a chocolate store in district j "*covers*" S(j).

Choose a minimum number of subsets that cover all of the districts.

Set covering Problem

Let **S** = {1, 2, …, n}, and suppose $S_j \subseteq$ **S** for each j.

We say that index set J is a <u>*cover*</u> of **S** if

$$\bigcup_{j \in J} S_j = \mathbf{S}$$

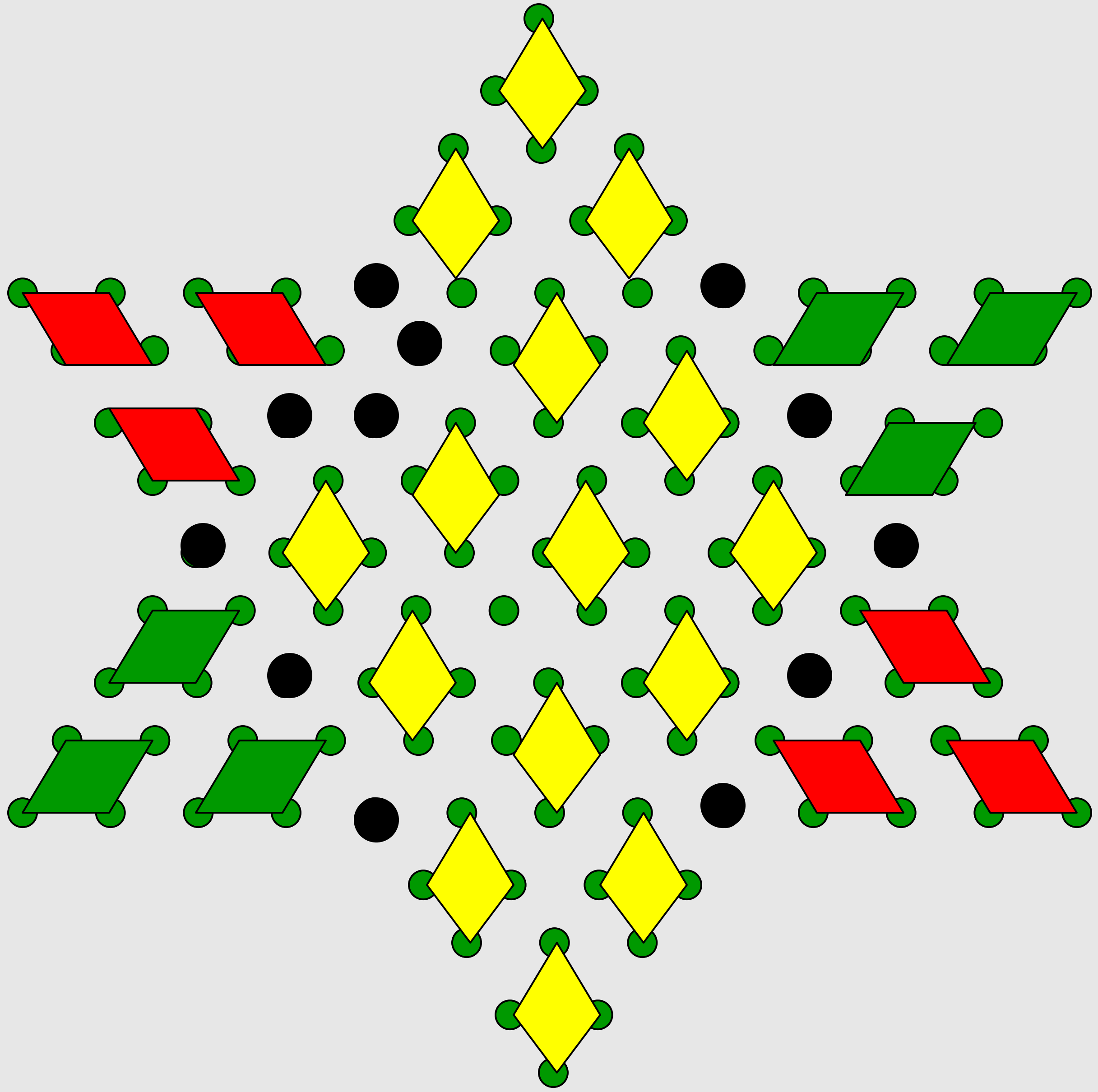
<u>*Set covering problem*</u>: find a minimum cardinality set cover of **S**.

Applications

- Locating fire stations.
- Locating hospitals.
- Locating Starbucks

and many non-obvious applications.

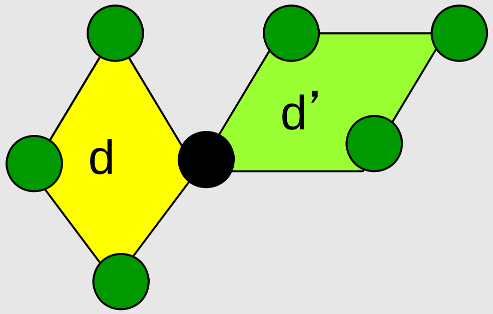
Packing diamonds into a Chinese checkerboard



The Diamond Packing Problem

- **Let D be the collection of diamonds**

- **Decision variables: x_d for $d \in D$**
  - **$x_d = 1$ if diamond d is selected**
  - **$x_d = 0$ if diamond d is not selected.**



$$x_d + x_{d'} \leq 1$$

Let O be the pairs of diamonds that overlap.
(d, d') ∈ O, implies that diamonds d and d' have at least one point in common

Set packing problem

Let $\mathbb{S} = \{1, 2, \ldots, n\}$, and suppose $S_j \subseteq \mathbb{S}$ for each j.

We say that index set J is a *packing* of $\mathbb{S}$ if

$S_i \cap S_j = \emptyset \quad$ for $i, j \in J$.

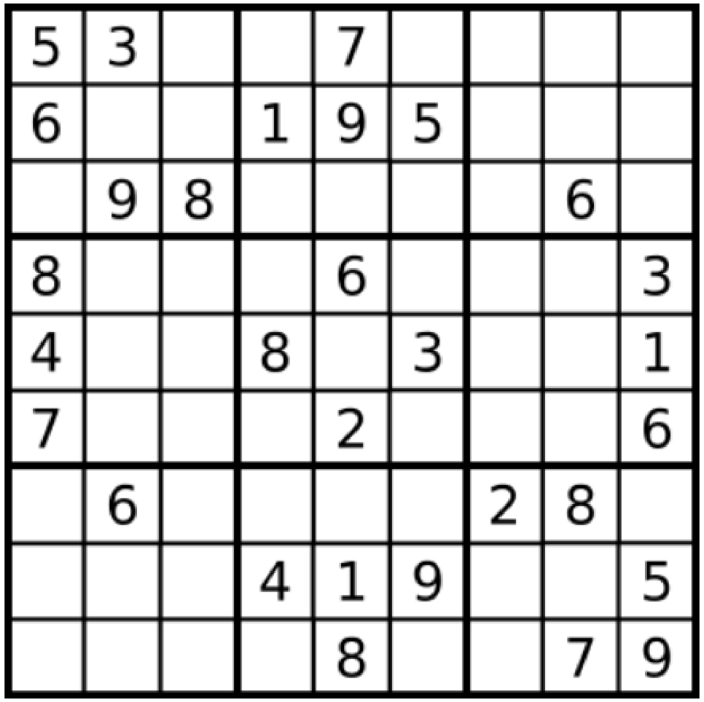
Set packing problem: find a maximum cardinality set cover of $\mathbb{S}$.

Common types of applications:

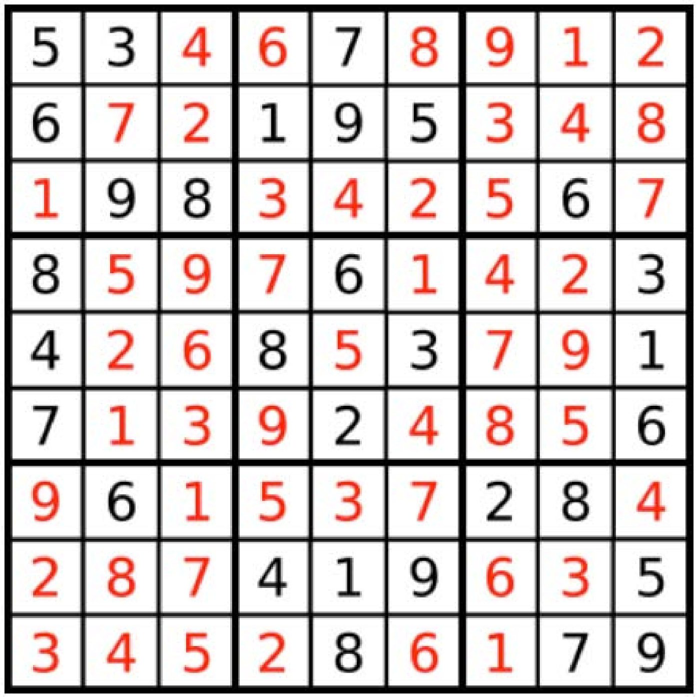
- Schedule a number of activities at the same time (activities cannot be scheduled at the same time if they require the same resource).
- Allocate building space

Formulating the game of Sudoku as an IP

- **Each row has each of the values in [1, 9]**
- **Each column has each of the values in [1, 9]**
- **Each of the nine 3 x 3 blocks has each of the values in [1, 9].**



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9



5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
2,1	2,2	2,3	2,4	2,5	2,6	2,7	2,8	2,9
3,1	3,2	3,3	3,4	3,5	3,6	3,7	3,8	3,9
4,1	4,2	4,3	4,4	4,5	4,6	4,7	4,8	4,9
5,1	5,2	5,3	5,4	5,5	5,6	5,7	5,8	5,9
6,1	6,2	6,3	6,4	6,5	6,6	6,7	6,8	6,9
7,1	7,2	7,3	7,4	7,5	7,6	7,7	7,8	7,9
8,1	8,2	8,3	8,4	8,5	8,6	8,7	8,8	8,9
9,1	9,2	9,3	9,4	9,5	9,6	9,7	9,8	9,9

Labels for cells in Sudoku

$$x_{ijk} = \begin{cases} 1 & \text{if the value in cell } (i, j) \text{ is } k \\ 0 & \text{otherwise} \end{cases}$$

9 x 9 x 9 = 729 variables.

Formulate these 9 constraints using integer programming

Each row has each of the numbers 1 to 9

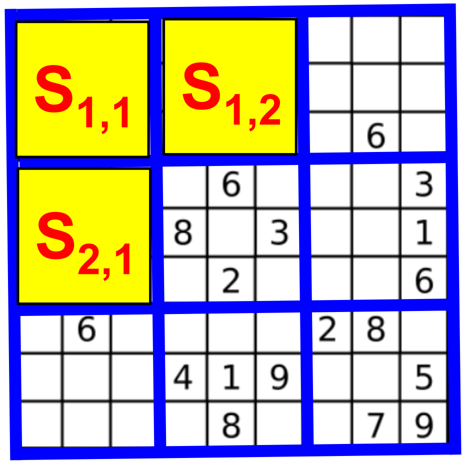
Formulate these 9 constraints using integer programming

Each column has each of the numbers 1 to 9

Don't bother with these yet. We need more notation

Each "block" has each of the numbers 1 to 9

Define 9 blocks within the 9 x 9 Sudoku game.



1,1	1,2	1,3
2,1	2,2	2,3
3,1	3,2	3,3

$S_{1,1}$ = {(1,1), (1, 2), (1, 3), (2, 1), …, (3, 3)}.

$S_{r,c}$ =

3r-2, 3c-2	3r-2, 3c-1	3r-2, 3c
3r-1, 3c-2	3r-1, 3c-1	3r-1, 3c
3r, 3c-2	3r, 3c-1	3r, 3c

for r = 1 to 3 and for c = 1 to 3.

$$\sum_{(i,j)\in S_{rc}}^{9} x_{ijk} = 1$$

for each r = 1 to 3, c = 1 to 3, and and for each k ∈ [1, 9]

What happens if an IP cannot be solved efficiently?

- Sometimes, an IP cannot be solved efficiently.

- Alternative approaches:
  - Use Branch and Bound but be less restrictive. Don’t ask for a guarantee of optimality. Be satisfied with 5% or 10% guarantees.
  - Use techniques from another field, such as artificial intelligence.
  - Rely on simple but practical heuristics for the problem at hand. (Illustrated in the 3rd lecture on networks.)

Final remarks on integer programming

- Very powerful modeling approach
  - Standard approach within math optimization
  - Can model virtually any combinatorial problem
  - Lots of techniques for modeling logical constraints and non-linearities

- Solution technique of choice: Branch and bound
  - works well if nodes can be pruned early
  - better bounding helps early pruning
  - better bounding via valid inequalities