# Integer Programming

- Pure Integer Programs: all decision variables restricted to integer values
  - 0-1 programs
  - general IP problems
- Mixed Integer Programs: some variables restricted to integer values; others may be continuous
  - 0-1 programs
  - general IP problems

Very versatile because many "real-world" applications involve variables that are naturally integer valued. More importantly, 0-1 variables allow us to model logical decisions

Also, many combinatorial optimization problems (where an optimum combination out of a possible set of combinations must be determined) can be cast as IPs.

©2020, Jayant Rajgopal

# *Examples of **0-1** Integer Programming Model Formulations*

**1.** LIMITED SET OF DISCRETE VALUES

Suppose $X$ can only take on one of the values in the set $\{a_1, a_2, \dots, a_n\}$; e.g., in designing a water distribution system the pipe diameter may be a variable that can belong to the set {6", 8", 10", 12"} because pipe is available in only these diameters. How do we model this?

**2.** KNAPSACK PROBLEMS

Suppose we have $n$ different items and item $i$ has weight $w_i$ and value $v_i$. A knapsack can hold at most $W$ units of weight. How do we load the knapsack so that we maximize the value of its contents?
In the multi-dimensional knapsack problem we may have additional constraints, e.g., we're given the volume of each item and the volume of the knapsack.

**3.** BATCH SIZE PROBLEMS

A variable in the model can be either zero or positive. However, if it is positive, it must be **at least** as large as some specified lower bound: e.g., the supplier for some raw material will only supply in amounts greater than or equal to some specific value. How do we model this?

**4.** BIN PACKING PROBLEMS (one dimensional)

Suppose we have $n$ different items and item $i$ has weight $w_i$. A bin can hold at most $W$ units of weight. How do we find the minimum number of bins in which we can pack all $n$ items (assume that $W \geq w_i$ for all $i$)?

**5.** FIXED CHARGE PROBLEMS

There is a fixed cost associated with an activity *regardless of the level of the activity* (usually in addition to some variable cost that *does* depend on the level of the activity…). For example, (1) if a set of items is purchased from a vendor, one incurs some fixed cost for order preparation, regardless of how many units are ordered, (2) when a batch of parts is made on a machine there is a fixed setup cost independent of the batch size, (3) when a production or distribution facility is operated, there may be a fixed cost that is independent of the volume handled at the facility, (4) when a crew is assigned to a task, there may be some fixed cost for the assignment regardless of the size of the crew, *et cetera*. All of these are example of the so called *fixed charge* problem.

**6.** PLANT LOCATION PROBLEMS

A simple example:
- $n$ sites in a region that require a product
- demand in area containing site $i = d_i$ units
- Want to erect no more than $m$ plants
- Capacity at site $i = s_i$
- $f_i$ cost to build and operate plant at site $i$
- $c_{ij}$ to ship each unit along route $i$-$j$

Problem: Find the optimal plant locations and the optimal shipping schedule to minimize all costs.
What if there is a cost $v_{ij}$ to set up and operate shipping route from $i$ to $j$? What if we don't have capacity constraints

**7.** CAPITAL BUDGETING PROBLEMS

Suppose a total of $m$ different choices of investments are available for selection: examples might include choosing among possible product lines for manufacture, selecting from different configurations of capital equipment, settling upon research and development projects, choosing a portfolio of investments etc. Often it makes no sense to go for partial investments and so each choice is a ***go / no-go*** option. Assume that $r_j$ is the return resulting from the $j^{th}$ investment and that $a_{ij}$ is the amount of resource $i$ (such as cash or manpower or space) that is used by the $j^{th}$ investment; there are $b_i$ units of resource $i$ available. Which investments should be chosen?

Extensions: Incorporating other logical considerations:
*"Either-Or" conditions*: Either investment $k$ or investment $l$ can be selected, but not both
*"No more than" conditions*: At most $p$ investments can be selected (where $p<m$).
*"At least" conditions*: At least $p$ investments must be selected (where $p<m$).
*"If-Then" conditions*: a) If investment $k$ is chosen, then investment $l$ must also be chosen.
b) If investment $k$ is chosen, then investment $l$ cannot be chosen.
*Multiple choice constraints*: Say we group the investments into different classes/types. We want at least one from each class to be selected. Or perhaps, the constraint is that IF we choose a particular class of investments then we want exactly one from this class.

## **8.** SET COVERING, PACKING AND PARTITIONING PROBLEMS

Let $\Gamma$ be some given set of $n$ items and $\boldsymbol{B}=\{\boldsymbol{B_1},...,\boldsymbol{B_m}\}$, where each $\boldsymbol{B_j}$ is a nonempty subset of $\Gamma$ with $\boldsymbol{B_1}\cup\boldsymbol{B_2}\cup\ldots\cup\boldsymbol{B_m}=\Gamma$. Also define for each $\boldsymbol{i=1,2,\ldots,n}$, the set $\boldsymbol{F_i = \{j\in(1,2,\ldots,m) \mid i\in B_j\}}$.

In a *set representation* problem we want to find some subset of $\Gamma$ **(say $E$)** such that $\boldsymbol{E}$ contains at least one common element with each subset $\boldsymbol{B_j}$ (i.e., a "representative" of that set). The objective might be to find the representation with the smallest number of elements, i.e., a *minimum cardinality representation* (MCR) for $\boldsymbol{B}$, or perhaps the least cost representation.

In a *set covering* problem we want to find a finite collection of subsets (say $\Omega$**)** from the set of subsets $\boldsymbol{B}=\{\boldsymbol{B_1},...,\boldsymbol{B_m}\}$, such that each element of the set $\Gamma$ is contained in at least one of the subsets selected for $\Omega$. Here $\Omega$ is called a "cover" for the set $\Gamma$ (some members in $\Gamma$ may appear in more than one subset in set $\Omega$). The objective might be to find the cover that has the minimum no. of subsets from $\boldsymbol{B}$ (i.e., minimum cardinality), or if there is a cost $c_j$ associated with a particular subset $\boldsymbol{B_j}$, we might want to find the minimum cost cover

The *set partitioning* problem is similar, except that each member of $\Gamma$ is assigned to *exactly one* subset in $\Omega$. In other words the members of $\Gamma$ are to be "partitioned" among members of set $\Omega$.

In a *set packing* problem, we try to choose some finite collection of subsets (say $\Omega$**)** from the set of subsets $\boldsymbol{B}=\{\boldsymbol{B_1},...,\boldsymbol{B_m}\}$, that are *mutually disjoint*. The objective could be to maximize the number of these subsets (i.e., $|\Omega|$**)** or the value of the subsets selected in the collection $\Omega$.

Example 1a: Consider the problem of locating a set of facilities (fire stations or hospitals or police stations…) to serve some collection $\Gamma$ of $\boldsymbol{n}$ different areas. Suppose we have $\boldsymbol{m}$ candidate locations for locating these facilities, and each location $j$ can service some subset $\boldsymbol{B_j}$ of areas from the set $\Gamma$. Our goal might to be find the smallest (or least cost) such set of facility locations so that all areas are covered. This may be formulate as a set covering problem.

Example 1b: Consider an airline flight personnel scheduling problem. The airline has a number of routing ***legs*** (e.g., 10 A.M. N.Y. to Chicago, or 6 P.M. Chicago to L.A.) to be flown. A combination of these legs constitutes a ***duty period***, and a sequence of duty periods that begins and ends at the same city is called a ***pairing*.** Crews must be then assigned to these pairings. Each pairing has several complex rules (e.g., no more than 7 flight legs, at least 9.5 hours between consecutive duty periods, each duty period can be no more than 12 hours etc.). The objective is to form a minimum cost set of pairings that covers all flight legs ($\Gamma$) in a time table.
One approach: Suppose we have a total of $\boldsymbol{n}$ flight legs to be covered, and that we can *generate* a total of $\boldsymbol{m}$ good pairings (where $\boldsymbol{m}>>\boldsymbol{n}$), where each pairing is made up of one or more legs (subsets of $\Gamma$). The pairings are constructed by taking into account arrival and departure times for making connections between two successive legs, allowing for ground maintenance times, other constraints such as those listed above, etc. Let $\boldsymbol{c_j}$ be the cost of assigning a crew to the legs in pairing $j$. We may then formulate this as a set covering problem with the objective of minimizing crew costs.

Example 2: Consider a region consisting of $\boldsymbol{n}$ different sales ***areas***. These sales areas are to be combined into sales ***regions*** and a sales rep is to be assigned to each region. Suppose we can generate a set of $\boldsymbol{m}$ potential regions where each region is a collection of some areas (which provides a good amount of work for the sales rep and perhaps satisfies other constraints). Let the cost of assigning a sales rep to the $j^{th}$ collection be $\boldsymbol{c_j}$. The problem may then be formulated as a set partitioning problem.

Example 3: Consider the problem of scheduling a number of meetings involving a number of different people. Suppose $\boldsymbol{m}$ meetings have to be scheduled next week; for the sake of simplicity, assume that each meeting lasts 1 hour and that there are $\boldsymbol{T}$ different one hour time slots available during the next week. Suppose that there are a total of $\boldsymbol{n}$ people and we are given the list of meetings from the collection of $\boldsymbol{m}$ meetings that each of the $n$ people must attend ($\boldsymbol{F_i}$). The problem of scheduling as many meetings as possible without any conflicts maybe formulated as a set packing problem.

# Set Covering/Partitioning/Packing

- $\Gamma$**={A,B,C,D,E,F,G,H,I,J,K}**

| | | |
|---|---|---|
| ***B*=** | **B1** =(A,B,C,D)<br>**B2** =(B,C,F,G)<br>**B3** =(A,H)<br>**B4** =(B,D,E,I,J)<br>**B5** =(D,E,F,K) | **B6** =(A,C,J,K)<br>**B7** =(D,G)<br>**B8** =(A,H,J,K)<br>**B9** =(E,F,I)<br>**B10** =(G,H,J,K) |

- $\mathbf{F_A}$**=(B1,B3,B6,B8);** $\mathbf{F_B}$**=(B1,B2,B4);** $\mathbf{F_C}$**=(B1,B2,B6); …;** $\mathbf{F_H}$**=(B3,B8,B10); …;** $\mathbf{F_K}$**=(B5,B6,B8,B10);**

- Set Representation: **E=** (A,G,I,K) or (A,E,G) or (A,B,D,E,K)

- Set Covering: $\Omega$**=** (B2,B4, B8) or (B1,B2,B8,B9) or (B1,B3,B5,B7,B8,B9)
- Set Partitioning: $\Omega$**=** (B1,B9,B10)
- Set Packing: $\Omega$**=** (B6,B7,B9) or (B1,B9,B10)

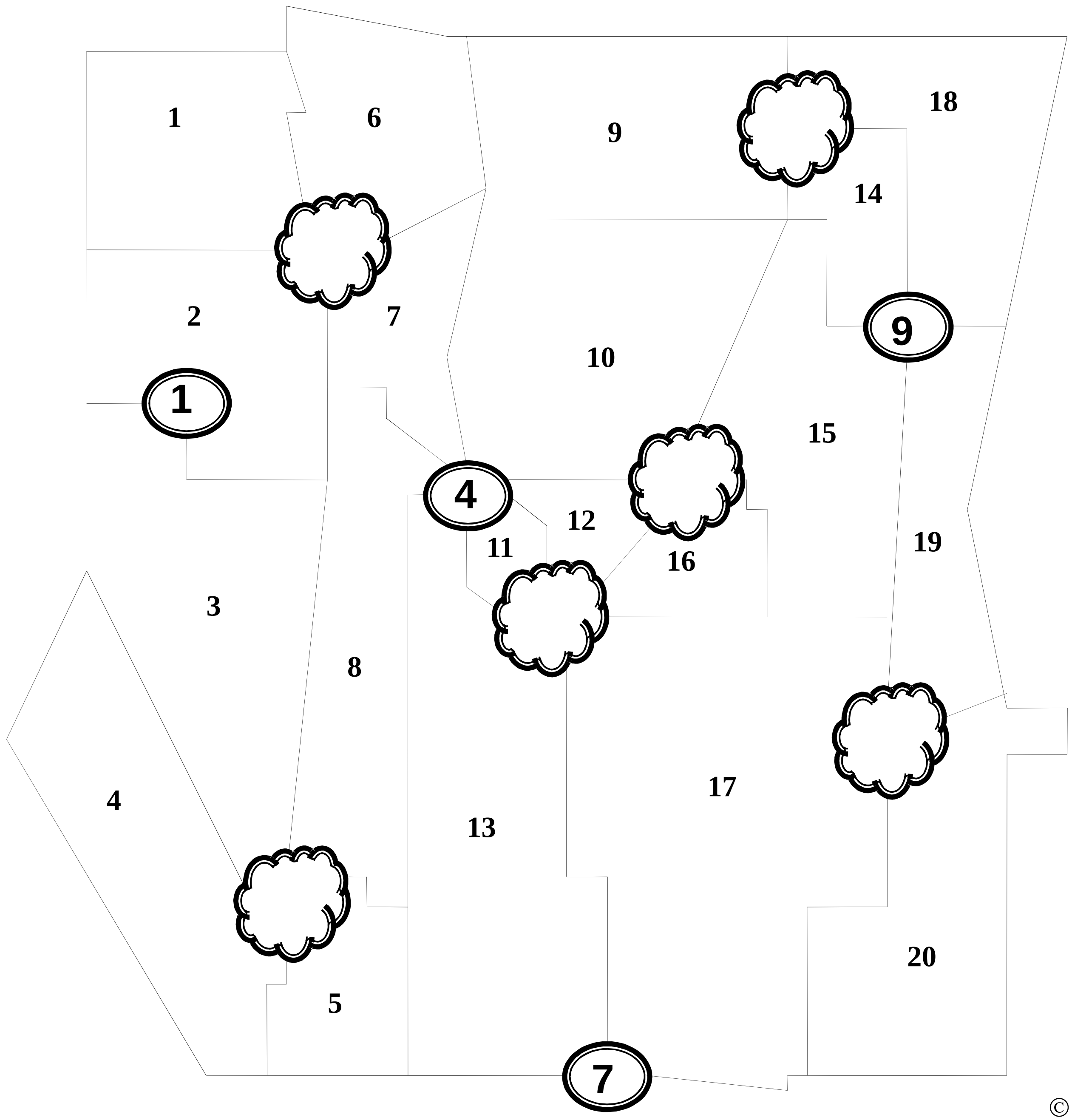
© 2020, Jayant Rajgopal

# Examples of Set Covering/Partitioning/ Packing Problems

| **Application** | $\Gamma$ | $B = \{B_1, B_2, \ldots, B_m)$ | $F_i$ | $X_j=1$ *if…* |
|---|---|---|---|---|
| Delivery & Routing | Set of *n* delivery locations | Set of *m* routes, each covering some subset of locations | Routes that contain location *i* | …route *j* is chosen |
| Facility Location | Set of *n* areas that require service | Set of *m* locations, each servicing some subset of areas | Locations that service area *i* | …location *j* is chosen |
| Fire Hydrant Location | Set of *n* street blocks | Set of *m* locations, each servicing some subset of street blocks | Locations that service block *i* | …location *j* is chosen |
| Sales force Assignment | Set of *n* sales areas to be covered | Set of possible assignments, each one covering some subset of areas | Assignments that cover area *i* | …assignment *j* is chosen |
| Crew Scheduling | Set of *n* flight legs or segments | Set of *m* pairings each covering some subset of (sequential) flight legs | Pairings that cover flight segment *i* | …pairing *j* is chosen |
| Committee Selection | Set of *n* desired committee characteristics | Set of groups of people (perhaps based on feasibility or availability) each having some subset of characteristics | Groups that contain characteristic *i* | …group *j* is chosen |

© 2020, Jayant Rajgopal

# Service Districts and Candidate Locations for EMS




© 2020, Jayant Rajgopal

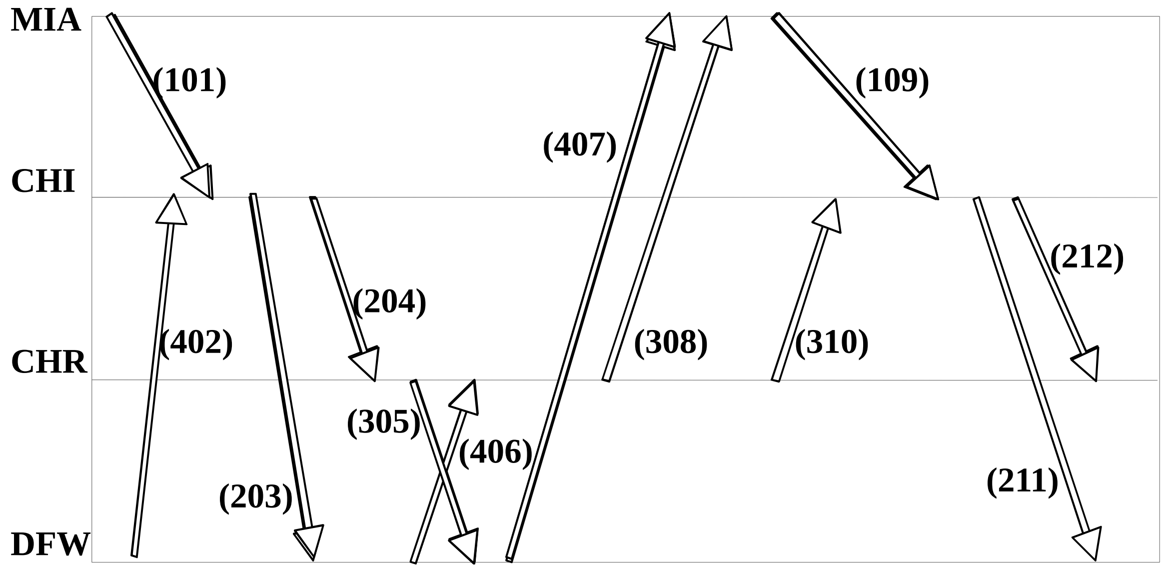



FIGURE: Flight Schedules for AA Example

TABLE: Possible Pairings for AA Example

| *j* | Flights Sequence | Cost | *j* | Flights Sequence | Cost |
|---|---|---|---|---|---|
| 1 | 101-203-406-308 | 2900 | 9 | 305-407-109-212 | 2600 |
| 2 | 101-203-407 | 2700 | 10 | 308-109-212 | 2050 |
| 3 | 101-204-305-407 | 2600 | 11 | 402-204-305 | 2400 |
| 4 | 101-204-308 | 3000 | 12 | 402-204-310-211 | 3600 |
| 5 | 203-406-310 | 2600 | 13 | 406-308-109-211 | 2550 |
| 6 | 203-407-109 | 3150 | 14 | 406-310-211 | 2650 |
| 7 | 204-305-407-109 | 2550 | 15 | 407-109-211 | 2350 |
| 8 | 204-308-109 | 2500 | | | |

© 2020, Jayant Rajgopal

**9.** USING 0-1 VARIABLES FOR OTHER LOGICAL CONSTRAINTS

Binary variables can be used to define a variety of logical constraints involving any set of variables; this usually results in a mixed integer program involving the usual continuous variables as well as 0-1 variables.

Constraint Feasibility: The basis for all logical constraints arises from answering the following question: how does one use a 0-1 variable in order to distinguish between when a constraint of the form $g(x_1,x_2,...,x_n) \geq 0$ (or $g(x_1,x_2,...,x_n) \leq 0$) is *definitely* satisfied, and when it may or may not be satisfied?

Alternative Constraints: Suppose a problem has two constraints of the form $g_1(x_1,x_2,...,x_n) \leq 0$ and $g_2(x_1,x_2,...,x_n) \leq 0$, but we only require that ***at least*** one of the two must hold. How would you account for this ? How would you extend the same logic if you had $m$ constraints out of which ***at least*** $k$ should hold?
E.g., we want to enforce $0 \leq x_1, x_2 \leq 10$ and at least one of $x_1$ and $x_2$ must be $\leq 5$.

Conditional Constraints: Suppose you have constraints of the form $g_1(x_1,x_2,...,x_n) > 0$ implies that $g_2(x_1,x_2,...,x_n) \leq 0$. How would you model this ?
(Note that the ***only circumstance*** under which the above implication is not satisfied is when both $g_1(x_1,x_2,...,x_n)>0$ and $g_2(x_1,x_2,...,x_n)>0$).

Compound Alternatives: Suppose the set of $n$ constraints are partitioned into $k$ subsets as

$S_1 = \{g_i(x_1,x_2,...,x_n) \leq 0 \text{ for } i=1,2,..n_1\}$,
$S_2 = \{g_i(x_1,x_2,...,x_n) \leq 0 \text{ for } i=n_1+1,n_1+2,..n_2\}$,
: : :
$S_k = \{g_i(x_1,x_2,...,x_n) \leq 0 \text{ for } i=n_{k-1}+1,n_{k-1}+2,..n\}$

How would you model the requirement that ***at least one set of the constraints*** should be satisfied?

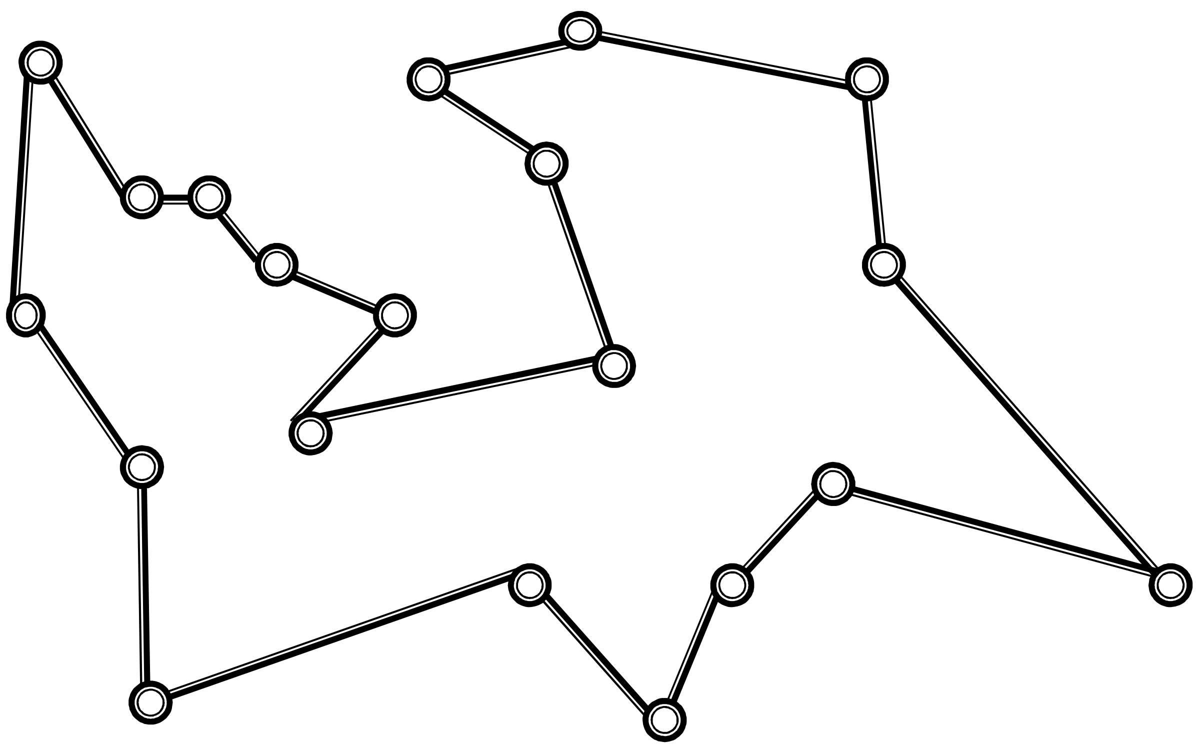
**10.** TRAVELING SALESMAN PROBLEM (a node covering problem):
Link ***i-j*** costs $c_{ij}$ for $i \neq j$. Find the cheapest path that starts at some city $i$ and returns there after visiting each city exactly once.

**11.** PIECEWISE LINEAR FUNCTIONS
Suppose a function $f(x)$ is of the form

- $m_1x$ $\qquad 0 \leq x \leq b_1$
- $m_1b_1 + m_2x$ $\qquad 0 \leq x \leq b_2$
- $m_1b_1 + m_2b_2 + m_3x$ $\qquad 0 \leq x \leq b_3$
- $m_1b_1 + m_2b_2 + m_3b_3 + m_4x$ $\qquad 0 \leq x \leq b_4$

Can we use LP to solve such problems?

# The Traveling Salesman Problem



- Widely studied problem
- Very hard to solve larger instances (NP-complete)
- Instances with >5000 cities have been solved optimally
- *Very* large instances have been solved approximately (10 million cities to within a couple of percent of optimum)

- What is a minimum length tour that visits all nodes?

© 2020, Jayant Rajgopal

# IP Approach

Let $x_{ij}$ = 1 if arc $i$-$j$ is in the tour
0 otherwise

Minimize $\sum_i \sum_j c_{ij} x_{ij}$

subject to

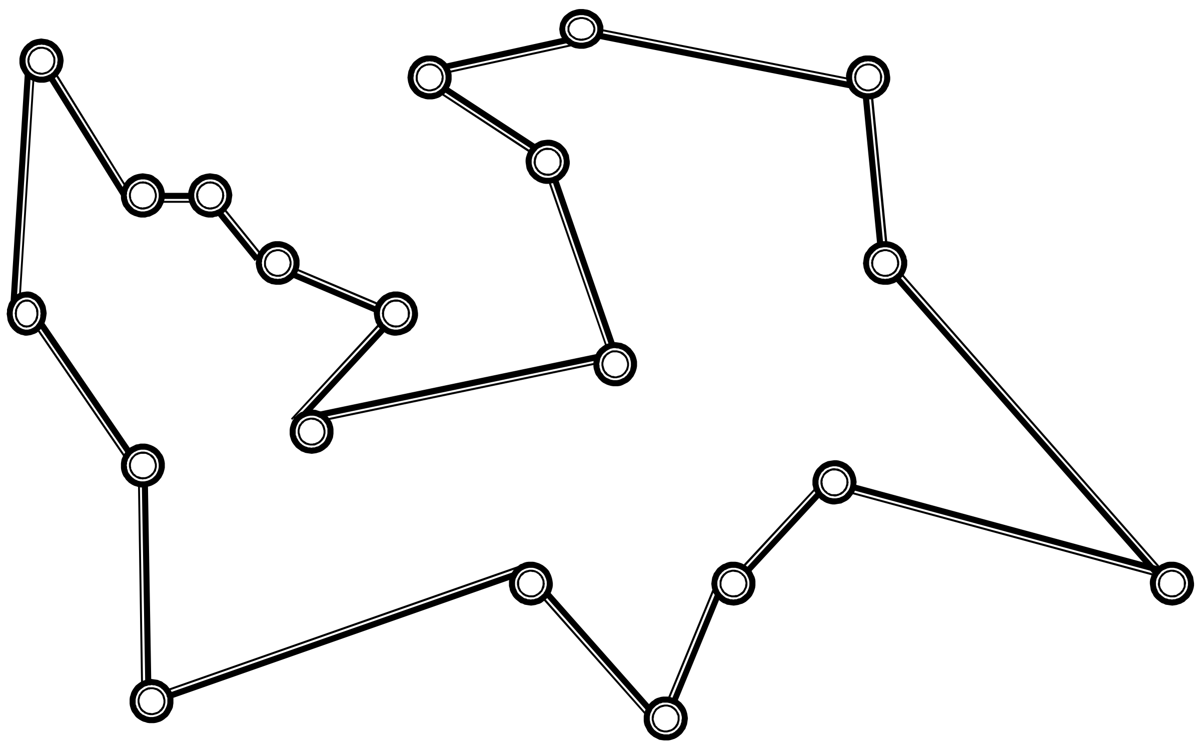
$$\sum_i x_{ij} = 1 \; \forall j$$

$$\sum_j x_{ij} = 1 \; \forall i$$

$$x_{ij} \in (0,1)$$

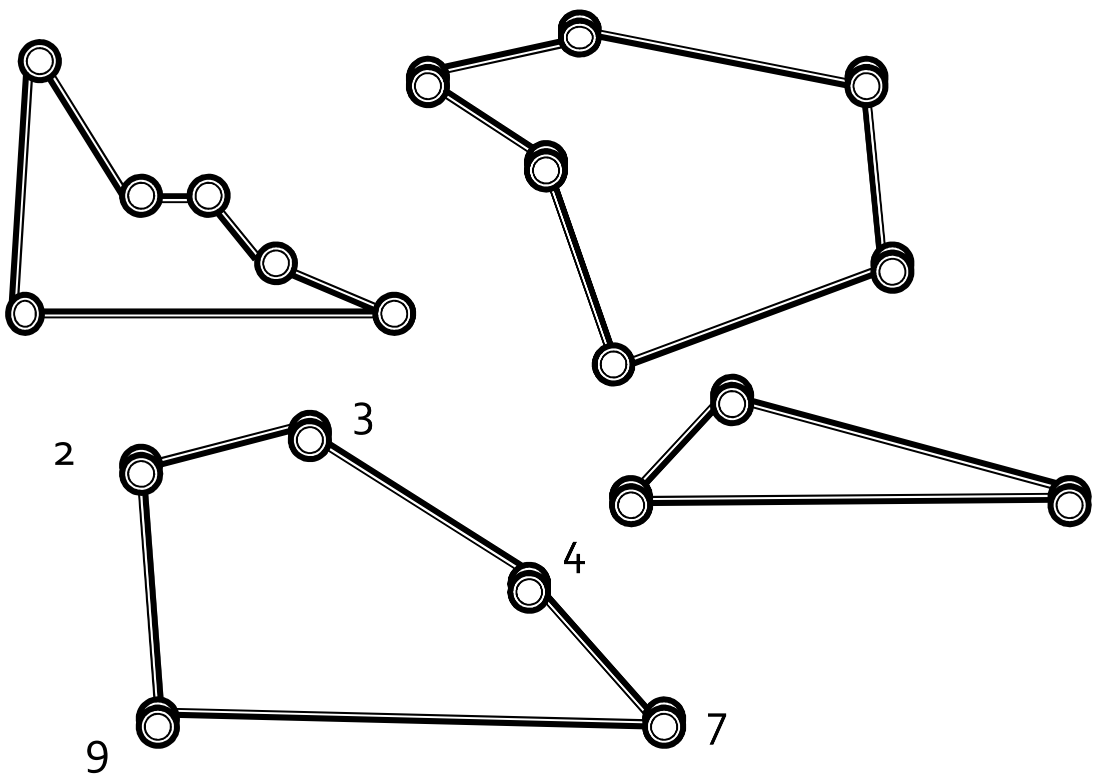
**Are these constraints enough?**

© 2020, Jayant Rajgopal



© 2020, Jayant Rajgopal

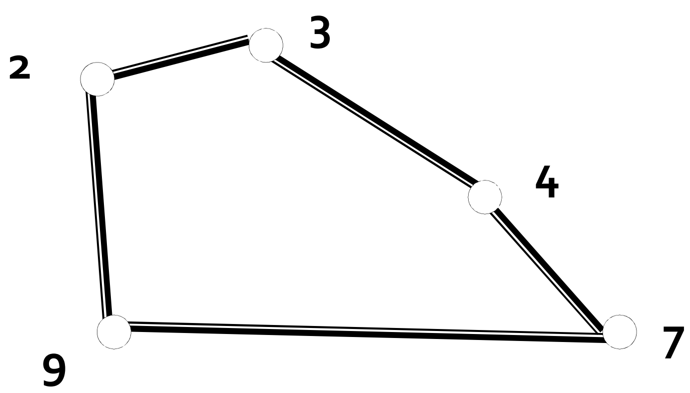
# *Subtour*: a cycle that passes through a strict subset of cities




Need to prevent subtours: for each possible subtour, add a constraint that makes the subtour infeasible for the IP.  These are called subtour breaking constraints.

© 2020, Jayant Rajgopal

# A subtour breaking constraint




Let S be any proper subset of nodes, e.g., S = {2, 3, 4, 7, 9}.

Observations:

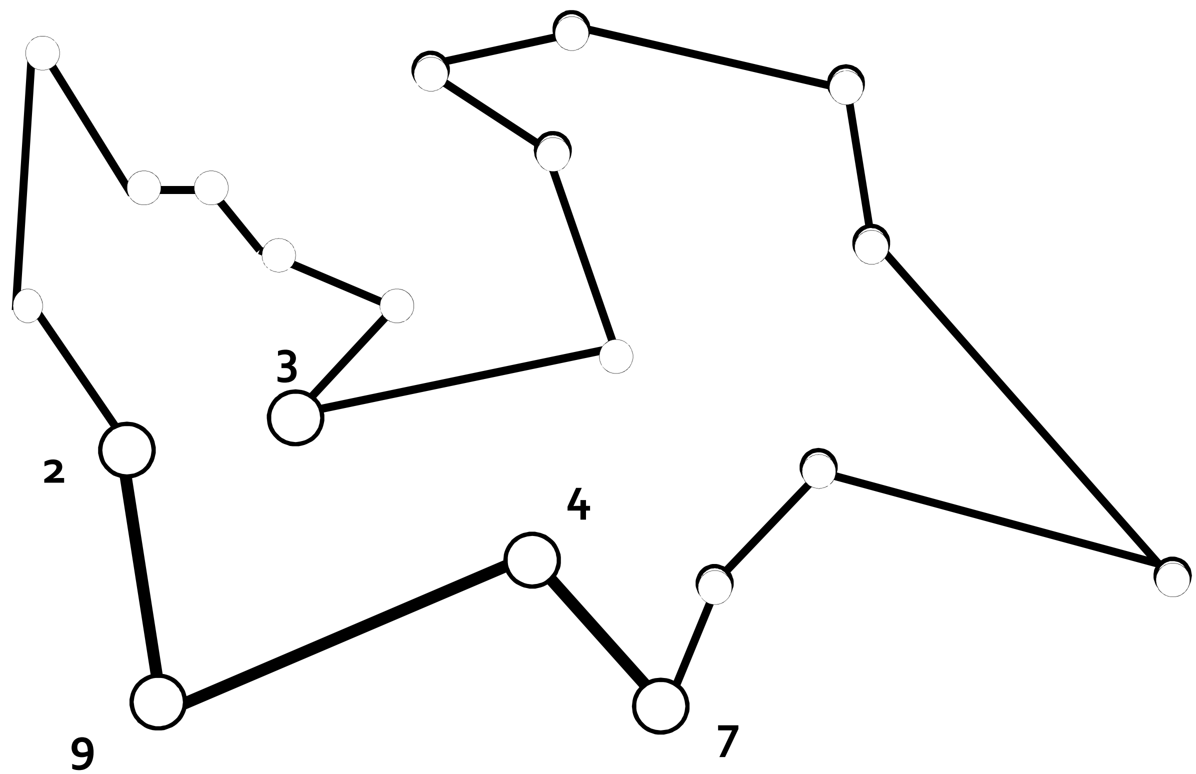
1. A subtour that includes all nodes of S has |S| arcs
2. A tour for the entire network has at most |S| - 1 arcs with two endpoints in S.

$$\sum_{i \in S, j \in S} x_{ij} \leq |S| - 1$$

**This ensures that the set S will not have a subtour going through all S nodes.**

© 2020, Jayant Rajgopal

# A traveling salesman tour




We constrain the problem so that there are at most 4 arcs in any tour incident to nodes 2, 3, 4, 7, 9. In this instance we have 3.

© 2020, Jayant Rajgopal

# Formulation of the TSP as an IP

Minimize $\sum_i \sum_j c_{ij} x_{ij}$

subject to

$$\sum_i x_{ij} = 1 \;\forall\, j$$

$$\sum_j x_{ij} = 1 \;\forall\, i$$

$$\sum_{i \in S, j \in S} x_{ij} \le |S| - 1$$

(subtour breaking constraints)

$$x_{ij} \in (0,1)$$

Exponentially many constraints, too many to include in an IP or an LP!

In practice:

- Include only some of the constraints. Solve the LP.
- If a subtour appears as part of the LP solution, add a new subtour elimination constraint to the LP, and solve again.

© 2020, Jayant Rajgopal

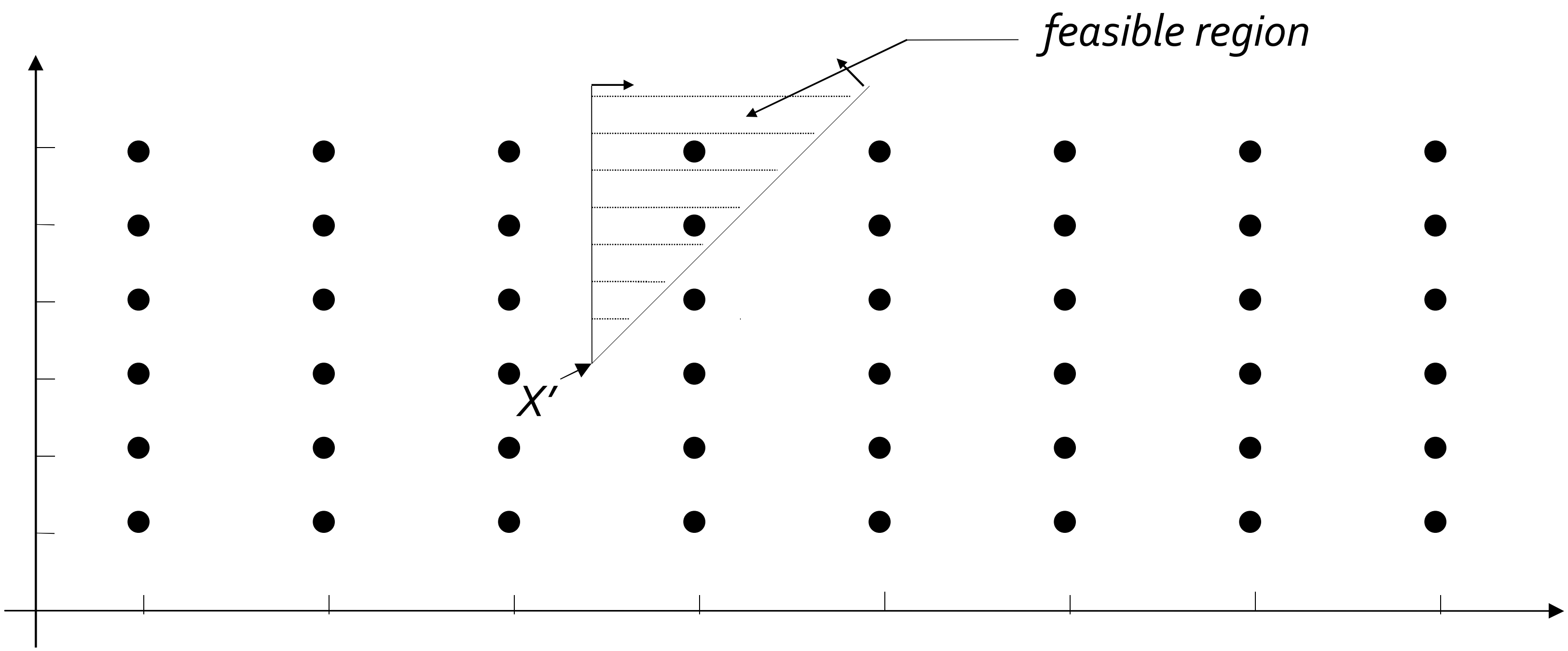
# Rounding Off…

Why not solve the Integer LP as a regular LP and "round off intelligently?"

- Not an illogical approach when all variables must be general integers. May turn out to be a practical approach, especially if all the integer variables are expected to be "large" at the optimum and feasibility is easy to maintain when rounding off

However…

- Usually doesn't make sense with 0-1 programs
- Many times all "rounded-off" solutions may be infeasible:

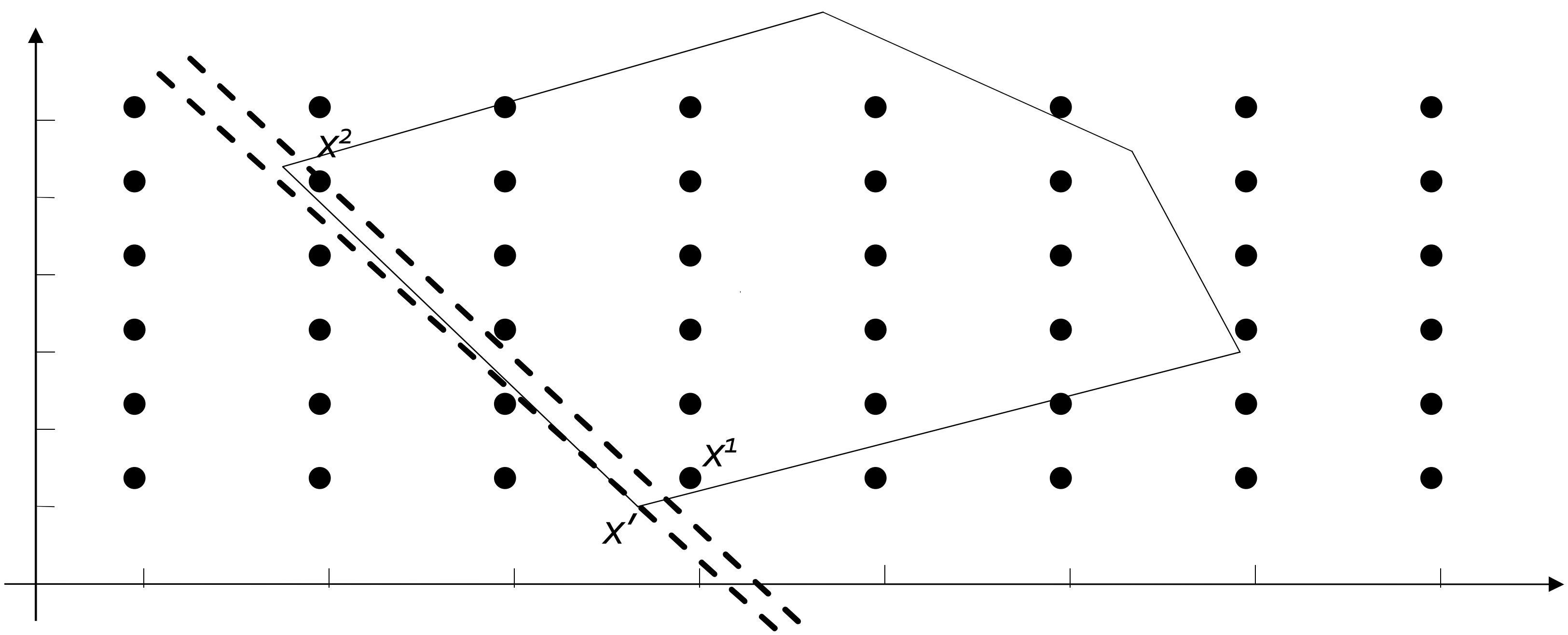



© 2020, Jayant Rajgopal

# Rounding Off…

- In other cases, the rounded solution might be quite far away from the integer optimum

*LP optimum = x′, rounded-off to $x^1$; whereas (the true) IP optimum= $x^2$ !!*




© 2020, Jayant Rajgopal