Integer Programming

(整数规划)

孙小玲教授
(xls@fudan.edu.cn)
复旦大学管理学院

时间：2012.7.23-2012.8.26
地点：台湾交通大学

Lecture 1: Introduction

(3 units)

Outline

- Course introduction
- What is integer programming?
- Simple examples
- Classification of integer programming problems
- Course description
- Examples from combinatorial optimization

Course introduction

- 32 units (45 minutes per unit) in 5 weeks.
- There will be 4 assignments.
- Scores will be based on the quality of the assignments.
- Course website:
  `http://my.gl.fudan.edu.cn/teacherhome/xlsun`

What is integer programming?

- Operations Research (O.R.) is the discipline of applying advanced analytical methods to help make better decisions. *Science of Better*.
- Mathematical Programming (Optimization) is a branch of OR. It is about how to make decision (with regard to some criteria) from some set of available alternatives.
- Integer Programming is about decision making with discrete or integer variables.

Why bother to use integer decision variable?

- If the variable is associated with a physical entity that is indivisible, then it must be integer, for example, number of airplanes to produce, number of shares of stock (roundlot) ...
- If a "Yes" or "No" decision has to be made (0 or 1).
- In most of these cases the continuous approximation to the discrete decision is not accurate enough for practical purposes.

Simple Examples

Example 1 (Stone Problem).

Given n stones of positive integer weights (i.e., given n positive integers $a_1, \ldots, a_n$), check whether you can partition these stones into two groups of equal weight, i.e., check whether a linear equation

$$\sum_{i=1}^{n} a_i x_i = 0$$

has a solution with $x_i \in \{-1, 1\}$, $\forall i$.
(How to model it to an optimization problem?)

Example 2 (Knapsack Problem)

A burglar has a knapsack of size b. He breaks into a store that carries a set of items n. Each item has profit c_j and size a_j . What items should the burglar select in order to optimize his heist?
Let

$$x_j = \begin{cases} 1, & \text{Item } j \text{ is selected} \\ 0, & \text{Otherwise} \end{cases}$$

The problem can be modeled as a 0-1 integer program:

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^{n} a_j x_j \leq b, \\ & x_j \in \{0,1\}, j = 1, \ldots, n. \end{aligned}$$

Example 3 (Assignment Problem).

- n people available to carry out n jobs. Each person is assigned to carry out one job. The cost for person i to do job j is c_{ij}. The problem is to find a minimum cost assignment.

$$x_{ij} = \begin{cases} 1, & \text{if person } i \text{ does job } j \\ 0, & \text{otherwise} \end{cases}$$

- The problem can be formulated as:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{j=1}^{n} x_{ij} = 1, i = 1, \ldots, n \\ & \sum_{i=1}^{n} x_{ij} = 1, j = 1, \ldots, n, \ x_{ij} \in \{0, 1\}. \end{aligned}$$

Some Real-World Optimization Problems

- Train scheduling
- Airline crew scheduling
- Highway pavement maintenance and rehabilitation
- Production planing
- Electricity generation planning
- Telecommunications
- Cutting problem

Classification of Integer Programming

- We call the problem mixed integer programming due to the presence of continuous variables.
- In some problems x are allowed to take on values only 0 or 1. Such variables are called binary. Integer programs involving only binary variables are called binary integer programs (BIPs). ($x \in \mathbb{B}^n$: $\{0,1\}^n$, or $x \in \{-1,1\}^n$)
- Pure Integer Linear Programming:

$$(IP) \quad \min \ \{c^T x \mid Ax \leq b, x \in \mathbb{Z}_+^n\}.$$

- Mixed 0-1 Programming:

$$(MIP) \quad \min\{c^T x + h^T y \mid Ax + Gy \leq b, x \in \mathbb{B}^n, y \in \mathbb{R}_+^p\}.$$

- A general (nonlinear) integer programming problem can be formulated as:

$$
\begin{aligned}
(NLIP) \qquad & \min \ f(x) \\
& \text{s.t.} \ \ g_i(x) \leq b_i, \ \ i = 1, \ldots, m, \\
& \qquad x \in X,
\end{aligned}
$$

  where f and g_i, $i = 1, \ldots, m$, are real-valued functions on $\mathbb{R}^n$, and X is a finite subset in $\mathbb{Z}^n$, the set of all integer points in $\mathbb{R}^n$.

- Mixed-integer nonlinear programming problem

$$
\begin{aligned}
(MNLIP) \qquad & \min \ f(x, y) \\
& \text{s.t.} \ \ g_i(x, y) \leq b_i, \ \ i = 1, \ldots, m, \\
& \qquad x \in X, \ \ y \in Y,
\end{aligned}
$$

  where f and g_i, $i = 1, \ldots, m$, are real-valued functions on $\mathbb{R}^{n+q}$, X is a finite subset in $\mathbb{Z}^n$, and Y is a continuous subset in $\mathbb{R}^q$.

How Hard is Integer Programming?

- First thought (a bit naive):
  - Total enumeration: For Stone Problem and 0-1 Knapsack Problem. To list all the feasible points, a super computer with speed 10^8 (Yi) basic operations per second needs:
  - $n = 30$, $2^{30} \approx 10^9$, 10 seconds.
  - $n = 60$, $2^{60} \approx 10^{18}$, 360 years
  - $n = 100$, $2^{100} \approx 10^{30}$, 4×10^{14} years
  - ...

- Round off: Solve the continuous optimization and round the solution to its nearest integer points.
- Heuristic methods. Greedy and local search methods. For 0-1 knapsack problem. Ranking the ratio of the profit to weight as:

$$\frac{c_{j_1}}{a_{j_1}} \geq \frac{c_{j_2}}{a_{j_2}} \geq \cdots \geq \frac{c_{j_n}}{a_{j_n}}.$$

  Choose the items in the order $j_1, \ldots, j_n$, and stop when taking the next item to the bag exceeds the total capacity b.
- Optimal solution? (If not, construct an counterexample)

Deep thoughts

- Most of the integer programs are NP-complete or NP-hard, which means the problem is "as difficult as a combinatorial problem can be", if we knew an efficient algorithm for one problem, we would be able to convert it to an efficient algorithm to solve any other combinatorial problem.
- Solving the linear programming relaxation results in a lower bound on the optimal solution to the IP.
- Rounding to a feasible integer solution may be difficult or impossible.
- The optimal solution to the LP relaxation can be arbitrarily far away from the optimal solution to the MIP (except totally unimodular case).
- Solving general integer programs can be much more difficult than solving linear programs or convex optimization problems.
- This is more than just an empirical statement. There is a whole theory surrounding it (you will learn some soon)

Course objectives

- Be able to use integer variables for formulating complex mathematical models in management science, industrial engineering and transportation science and learn how to use various software package for solving integer programming models.
- Understand the concepts of NP-Completeness and NP-Hardness and the difficulty of integer programming problems using the tool of complexity theory.
- Be able to use the basic methodology for the solution of integer programs.
- Understand the basic concepts of polyhedral theory and valid inequalities and how to integrate the theory to the solution methods for integer programming.
- Understand the advanced methods based on column generation and Dantzig-Wolfe decomposition for large-scale integer programming.

Course Modules

- Module 1 (IP Basic)
  - Modeling problems with integer decision variables.
  - Branch-and-Bound method. The very basics of the "workhorse" algorithm for solving IPs
  - Software for IP (Matlab, CLPEX, AIMMS)
- Module 2 IP Theory
  - Complexity Theory. What makes a problem "hard" or "easy" The complexity classes P, and NP, and NP-completeness
  - Polyhedral Theory. Dimension, facets, polarity, and the equivalence of separation and optimization.

- Module 3 Basic Algorithms
  - Branch-and-bound
  - Dynamic programming
  - Cutting plane method
- Module 4 Advanced Algorithms
  - Lagrangian Relaxations
  - Column generation and Dantzig-Wolfe decomposition
  - Branch-and-Price.
- Module 5 Nonlinear IP
  - Quadratic 0-1 optimization, Max-cut problem
  - Quadratic knapsack problem

Combinatorial Optimization Problems

Example 1 (Set Covering Problem)

- Given a certain number of regions, the problem is to decide where to install a set of emergency service centers. For each possible center the cost of installing a service center, and which regions it can be service are known. The goal is to choose a minimum cost set of service centers so that each region is covered.
- Let $M = \{1, \ldots, m\}$ be the set of regions, and $N = \{1, \ldots, n\}$ the set of potential centers. Let $S_j \subseteq M$ be the regions that can be serviced by a center at $j \in N$, and c_j its installing cost.
- Define a 0-1 incidence matrix A such that $a_{ij} = 1$ if $i \in S_j$ and $a_{ij} = 0$ otherwise. Let

$$x_j = \begin{cases} 1, & \text{if center } j \text{ is selected} \\ 0, & \text{otherwise} \end{cases}$$

- The problem can be formulated as

$$\begin{aligned} \min & \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} & \sum_{j=1}^{n} a_{ij} x_j \geq 1, i = 1, \ldots, m \\ & x \in \{0,1\}^n. \end{aligned}$$

Example 2: Facility location problem

- Given a set $N = \{1, \ldots, n\}$ of potential facility locations and a set of clients $M = \{1, \ldots, m\}$. A facility placed at j costs f_j for $j \in N$. The profit from satisfying the demand of client i from a facility at j is c_{ij}. The problem is to choose a subset of the locations at which to place facilities, and then to assign the clients to these facilities so as to maximize the total profit.
- Let $x_j = 1$ if a facility is placed at j and $x_j = 0$ otherwise.
- Let y_{ij} be the fraction of the demand of client i that is satisfied from a facility at j.
- The problem can be modeled as as

$$\begin{aligned} \max & \sum_{i \in M} \sum_{j \in N} c_{ij} y_{ij} - \sum_{j \in N} f_j x_j \\ \text{s.t.} & \sum_{j \in N} y_{ij} = 1, \quad \forall i \in M, \\ & y_{ij} \leq x_j, \quad \forall i \in M,\ j \in N, \\ & x \in \{0,1\}^n,\ y \in \Re_+^{mn}. \end{aligned}$$

Example 3: Traveling Salesman Problem (TSP)

- A traveling salesman must visit each of n cities before returning home. He knows the distance between each of the cities and wishes to minimize the total distance traveled while visiting all of the cities.
- Problem: In what order should he visit the cities?

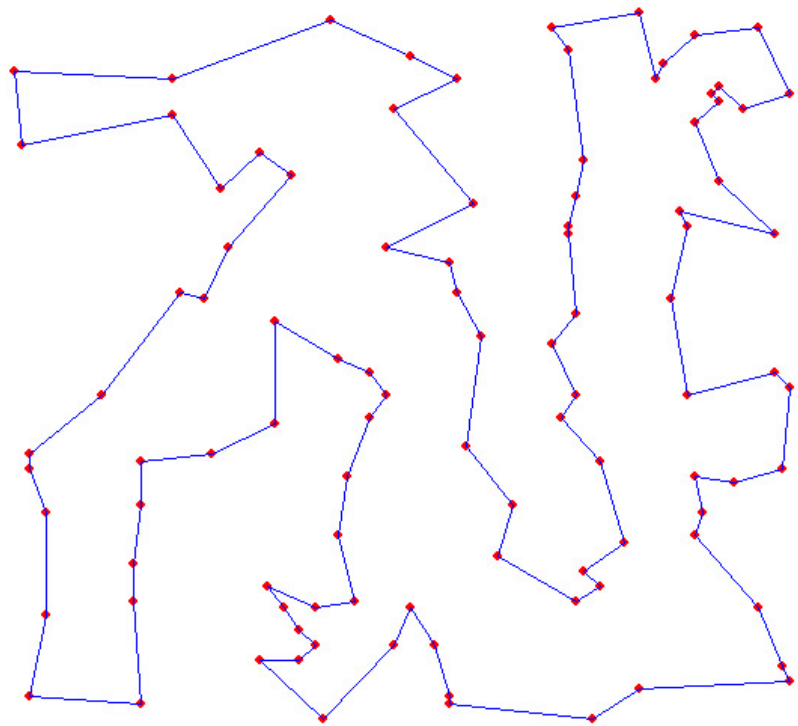


Figure: Traveling salesman problem

- Let the distance from city i to city j be c_{ij}. Let

$$x_{ij} = \begin{cases} 1, & \text{if he goes directly from } i \text{ to } j \\ 0, & \text{otherwise} \end{cases}$$

- Constraints:
  - He leaves city i exactly once:

$$\sum_{j \neq i} x_{ij} = 1, \ i = 1, \ldots, n.$$

  - He arrives at city j exactly once:

$$\sum_{i \neq j} x_{ij} = 1, \ j = 1, \ldots, n.$$

- Subtour elimination constraints:

$$\sum_{i\in S}\sum_{j\notin S} x_{ij} \geq 1, \forall S \subset N = \{1,\ldots,n\}, S \neq \emptyset,$$

or

$$\sum_{i\in S}\sum_{j\in S} x_{ij} \leq |S| - 1, \forall S \subset N, 2 \leq |S| \leq n-1.$$

- TSP can be formulated as

$$\begin{aligned}
\min \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \\
\text{s.t.} \quad & \sum_{j\neq i} x_{ij} = 1, \ i = 1,\ldots,n, \\
& \sum_{i\neq j} x_{ij} = 1, \ j = 1,\ldots,n, \\
& \sum_{i\in S}\sum_{j\in S} x_{ij} \leq |S| - 1, \forall S \subset N, 2 \leq |S| \leq n-1, \\
& x \in \{0,1\}^n.
\end{aligned}$$

Figure: TSP 1000 cities

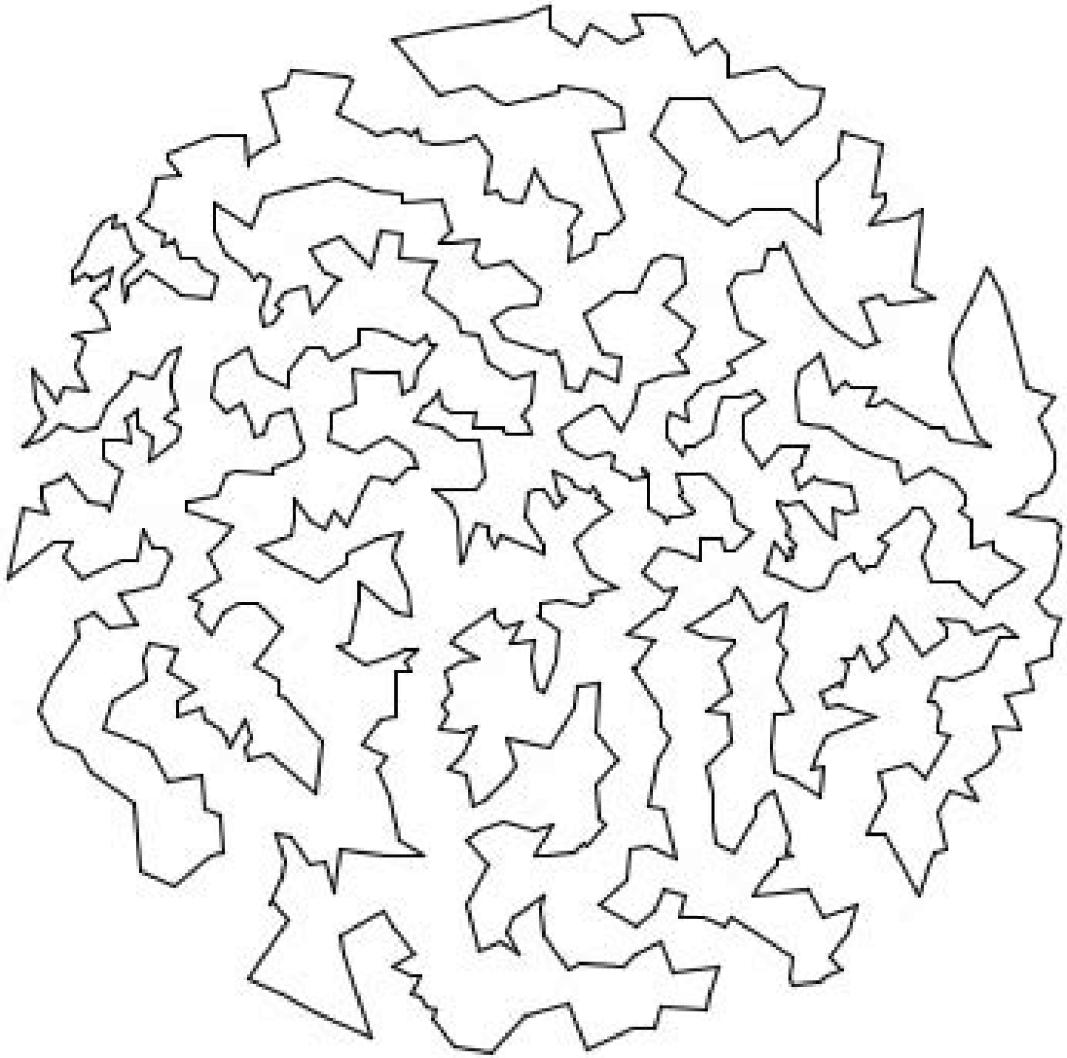


Figure: Optimal tour for the TSP 1000 cities

- A problem related to TSP is the vehicle routing problem: a number of vehicle located at a central depot has to serve a set of customers which geographically surround the depot. Each vehicle has a given capacity and each customer has a a given demand. The goal is to minimize the total travel distance.

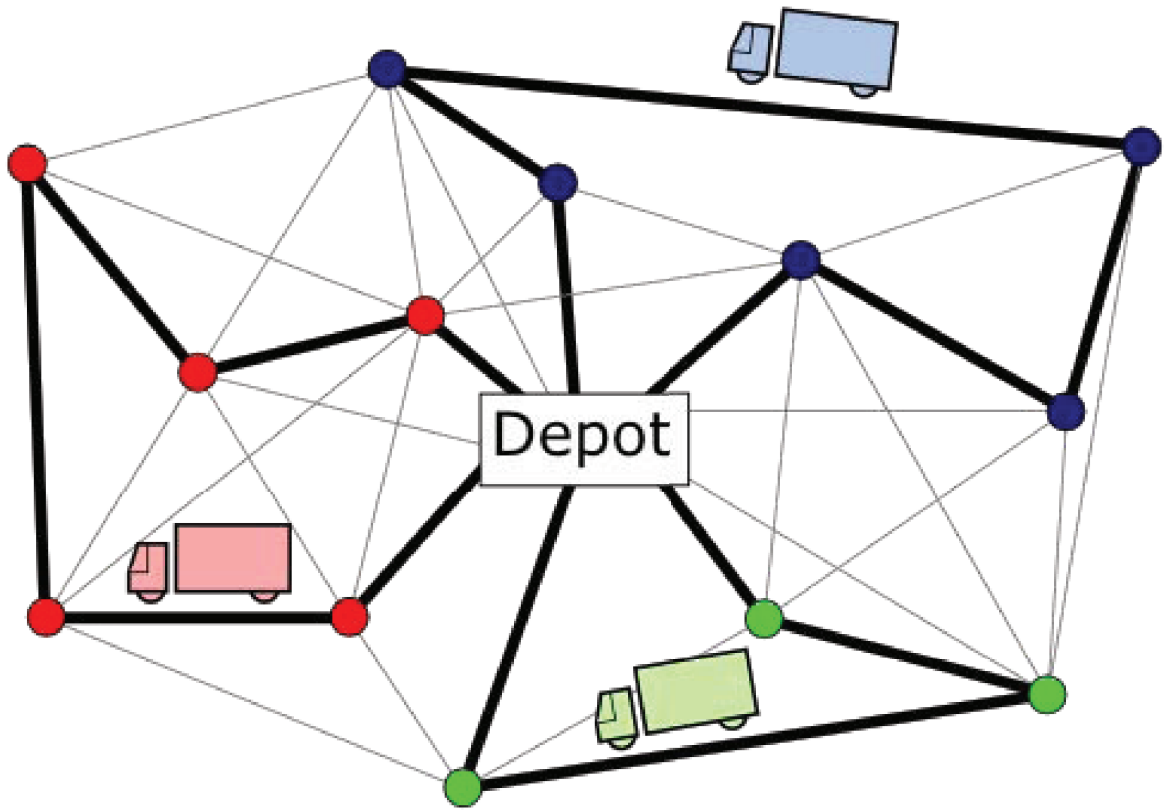


Figure: Vehicle routing problem

Example 4 (Generalized Assignment Problem)

Given m machines and n jobs, find a least cost assignment of jobs to machines not exceeding the machine capacities. Each job j requires a_{ij} units of machine i' s capacity b_i. The problem can be modeled as

$$\begin{aligned}
\min \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \\
\text{s.t.} \quad & \sum_{j=1}^{n} a_{ij}x_j \le b_i, \ \forall i, \quad \text{(machine capacity)} \\
& \sum_{i=1}^{m} x_{ij} = 1, \ \forall j, \quad \text{(assign all jobs)} \\
& x_{ij} \in \{0,1\}, \ \forall i,j
\end{aligned}$$

Example 5: Minimum cost network flows

- A digraph $G = (V, A)$ with arc capacity h_{ij} (integer), for $(i, j) \in A$, demand b_i at node $i \in V$, unit flow cost c_{ij} for $(i, j) \in A$.
- Find feasible integer flow with the minimum cost.
- x_{ij}: the flow for arc (i, j), $V_i^+ = \{k \mid (i, k) \in A\}$, $V_i^- = \{k \mid (k, i) \in A\}$.
- The model:

$$\begin{aligned} \min \quad & \sum_{(i,j)\in A} c_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{k\in V_i^+} x_{ik} - \sum_{k\in V_i^-} x_{ki} = b_i, \ \ i \in V \\ & 0 \le x_{ij} \le h_{ij}, x_{ij} \text{ integer}. \end{aligned}$$

Example 6: Shortest Path Problem

Given two nodes $s, t \in V$ and nonnegative arc costs c_{ij} for $(i, j) \in A$, find a minimum cost flow from s to t.

The problem can be formulated as

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in A} c_{ij} x_{ij} \\
\text{s.t.} \quad & \sum_{k\in V_i^+} x_{ik} - \sum_{k\in V_i^-} x_{ki} = 1, \ i = s \\
& \sum_{k\in V_i^+} x_{ik} - \sum_{k\in V_i^-} x_{ki} = 0, \ i \in V \setminus \{s, t\} \\
& \sum_{k\in V_i^+} x_{ik} - \sum_{k\in V_i^-} x_{ki} = -1, \ i = t \\
& x_{ij} \in \{0, 1\}, (i, j) \in A.
\end{aligned}
$$

Example 7: Maximum flow problem

- Given two nodes $s, t \in V$ and nonnegative arc capacities h_{ij} for $(i, j) \in A$, find a maximum flow from s to t.
- Adding a backward arc from t to s with $h_{ij} = +\infty$. The Maximum flow problem can be formulated as

$$\begin{aligned} \max \ & x_{ts} \\ \text{s.t.} \ & \sum_{k \in V_i^+} x_{ik} - \sum_{k \in V_i^-} x_{ki} = 0, \ i \in V, \\ & 0 \leq x_{ij} \leq h_{ij}, (i, j) \in A. \end{aligned}$$

Nonlinear Integer Programming Models

Example 1: Portfolio Optimization

- A market with n securities. An investor with initial wealth W_0 seeks to improve his wealth status by investing his wealth into these n risky securities and into a risk-free asset (e.g., a bank account).
- X_i: the random return per lot of the i-th security ($i = 1, \ldots, n$) before deducting associated transaction costs.
- The mean and covariance:

$$\mu_i = \mathrm{E}(X_i), \text{ and } \sigma_{ij} = \mathrm{Cov}(X_i, X_j), \qquad i, j = 1, \ldots, n.$$

- x_i: the integer number of lots the investor invests in the i-th security. Portfolio vector: $x = (x_1, \ldots, x_n)^T$.

- The random return: $P_s(x) = \sum_{i=1}^{n} x_i X_i$. The mean and variance of $P_s(x)$ are

$$s(x) = \mathrm{E}[P_s(x)] = \mathrm{E}[\sum_{i=1}^{n} x_i X_i] = \sum_{i=1}^{n} \mu_i x_i$$

and

$$\begin{aligned} V(x) &= \mathrm{Var}(P_s(x)) \\ &= \mathrm{Var}[\sum_{i=1}^{n} x_i X_i] \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j \sigma_{ij} = x^T C x, \end{aligned}$$

where $C = (\sigma_{ij})_{n\times n}$ is the covariance matrix.

- Let r be the interest rate of the risk-free asset.
- Transaction cost: $c(x) = \sum_{i=1}^{n} c_i(x_i)$.

- Total expected return of portfolio:

$$\begin{aligned} R(x) &= s(x) + rx_0 - \sum_{i=1}^{n} c_i(x_i) \\ &= \sum_{i=1}^{n} [(\mu_i - rb_i)x_i - c_i(x_i)] + rW_0. \end{aligned}$$

- Budget constraint:

$$b^T x \leq W_0 + U_b$$

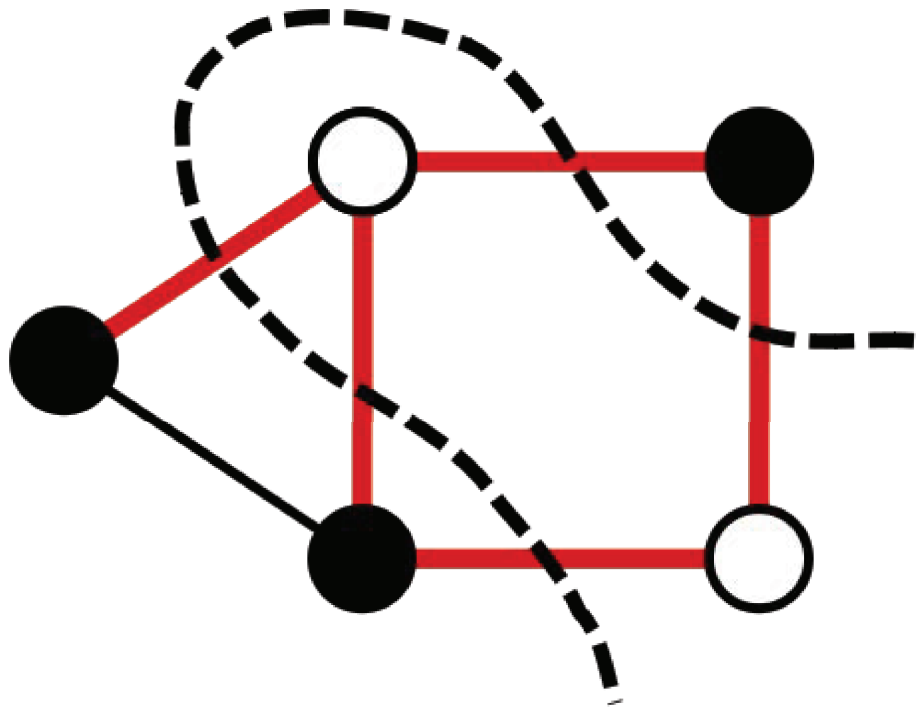
where $b = (b_1, \ldots, b_n)^T$ and U_b is the upper borrowing limit from the risk-free asset.

- Integer programming model:

$$\begin{aligned} (MV) \qquad & \min\ V(x) = x^T C x \\ & \text{s.t.}\ \ R(x) = \sum_{i=1}^{n} [(\mu_i - rb_i)x_i - c_i(x_i)] + rW_0 \geq \varepsilon, \\ & U(x) = b^T x \leq W_0 + U_b, \\ & x \in X = \{x \in \mathbb{Z}^n \mid l_i \leq x_i \leq u_i\}. \end{aligned}$$

Example 2: Maximum Cut

- Let G be an n-node graph, and let the arcs $(i; j)$ of the graph be associated with nonnegative "weights" a_{ij} $(a_{ij} = a_{ji} \geq 0)$.
- A cut (S, S') is a partition of the n nodes in two disjoint set.



- The problem is to find a cut of the largest possible weight, i.e., to partition the set of nodes in two parts S, S' so that the total weight of all arcs linking S and S' is maximized

- Let $x_i = 1$ for $i \in S$, $x_i = -1$ for $i \in S'$.
- Total weight of cut (S, S') is:

$$\frac{1}{2}\left(\frac{1}{2}\sum_{i,j=1}^{n} a_{ij}(1 - x_i x_j)\right) \text{ (Why?)}$$

$$= \frac{1}{4}\sum_{i,j=1}^{n} a_{ij}(1 - x_i x_j).$$

- The Maxcut problem can be modeled as

$$\max \ \frac{1}{4}\sum_{i,j=1}^{n} a_{ij}(1 - x_i x_j)$$

$$\text{s.t. } x \in \{-1, 1\}^n.$$

- Goemans and Williamson (1995)'s brilliant result on the quality of semidefinite program (SDP) relaxation of Maxcut problem:

$$0.878. \leq \frac{f_{opt}}{f_{SDP}} \leq 1$$

- Moreover, they showed that a 0.878. approximate (feasible) solution can be obtained by a randomized method to Maxcut problem!
(2000 AMS-MPS Fulkerson Prize)