Integer Programming

Instructor: Jin Zhang

Department of Mathematics
Southern University of Science and Technology
Spring 2024

What is Integer Programming?

- Integer Programming is about decision making with discrete or integer variables.

  Why bother to use integer decision variables?

  - 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 Example

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:

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

Example 3 (Assignment Problem)

- n people available to carry out n jobs. Each person 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
- Telecommunication planning
- Cutting problem

Classification of Integer Programming (1)

Decision variables

- All are integers: pure integer programming
- Integers (part) + continuous variables: Mixed integer programming
- $0-1$ variables $(-1, 1)$: binary integer programming
- $0-1$ variables (part) + continuous variables: Mixed binary integer programming

Classification of Integer Programming (2)

Objective and constraints

- Linear integer programming
- Nonlinear integer programming

Pure linear Integer programming:

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

Classification of Integer Programming (3)

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

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

  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:

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

  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 $10^8(\mathrm{Yi})$ basic operations per second needs:
  - $n = 30, 2^{30} \approx 10^9, 10$ seconds.
  - $n = 60, 2^{100} \approx 10^{18}, 360$ years
  - $n = 100, 2^{100} \approx 10^{30}, 4 \times 10^{14}$ years
  - $\ldots$

Approximate method

- 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)

Course objectives

- Modeling and software (primary objective)

  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.

- Exact algorithm (Branch and bound, cutting plane)

  Be able to use the basic methodology for the solution of integer programs.

- Advanced topics on algorithmic design will not be touched Approximate method, randomized method; Exact method: Column generation, Dantzig Wolf decomposition

Modeling (matrix language)

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 reformulated as:

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

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

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

Figure 1: The figure shows ICBC branches and geographic entities (e.g., bank branches designated by circled I, office buildings designated by gray dots, and residential buildings designated by black dots) in a typical city in China.

Cells

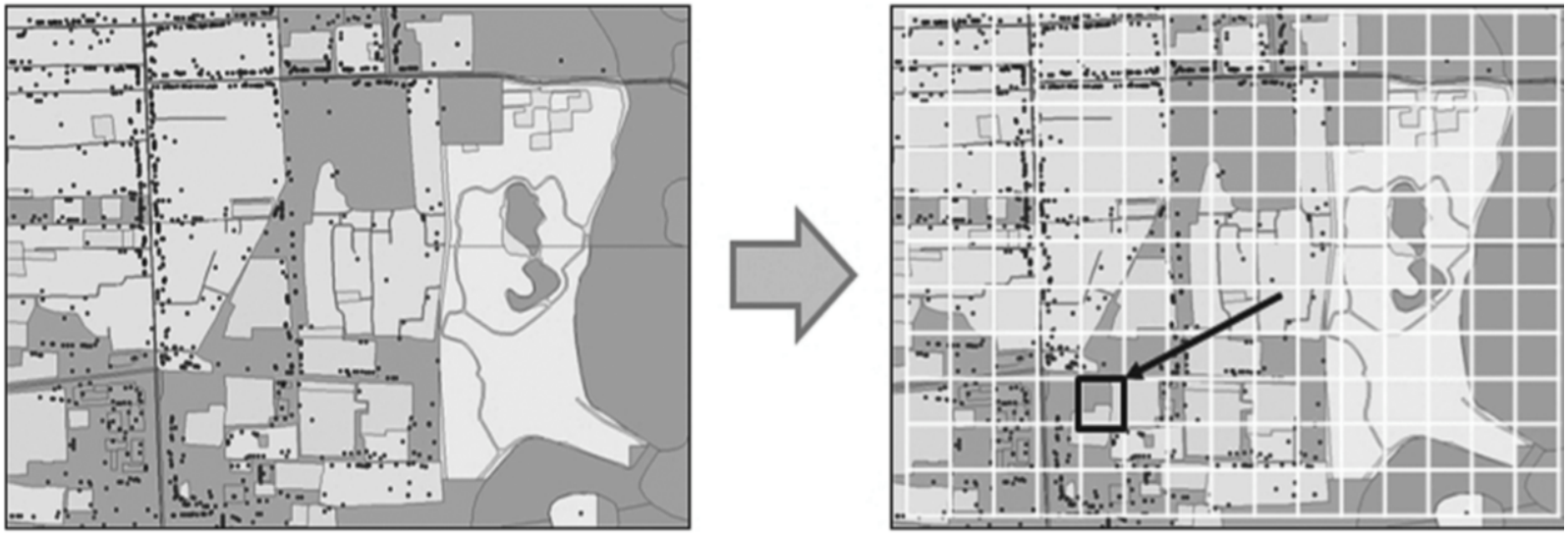


Figure A.1: BR divides a city area (left side of figure) into cells of 100 meters by 100 meters (right side of figure). The black box in the right figure represents such a cell.

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?

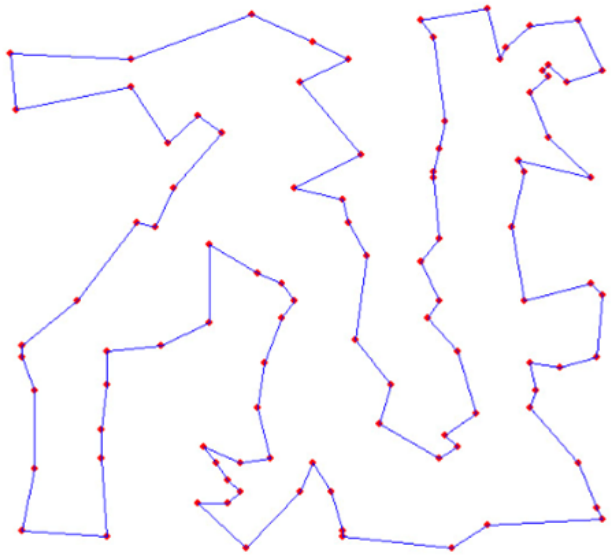


Figure: Traveling salesman problem

- 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

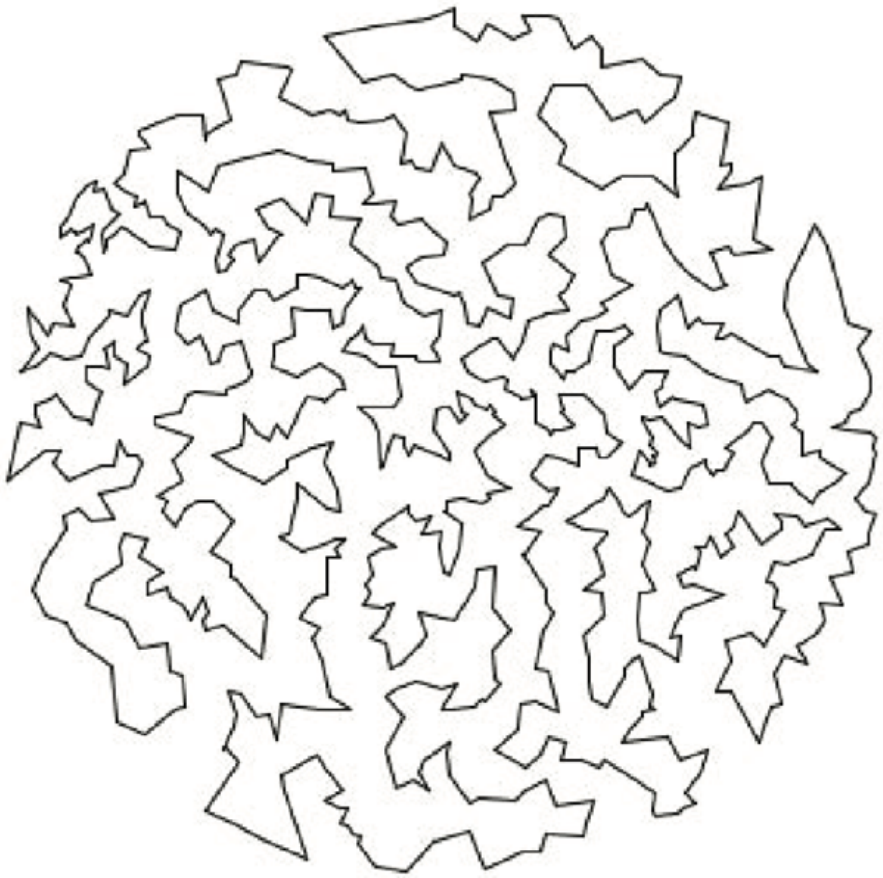


Figure: Optimal tour for the TSP 1000 cities

- 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.

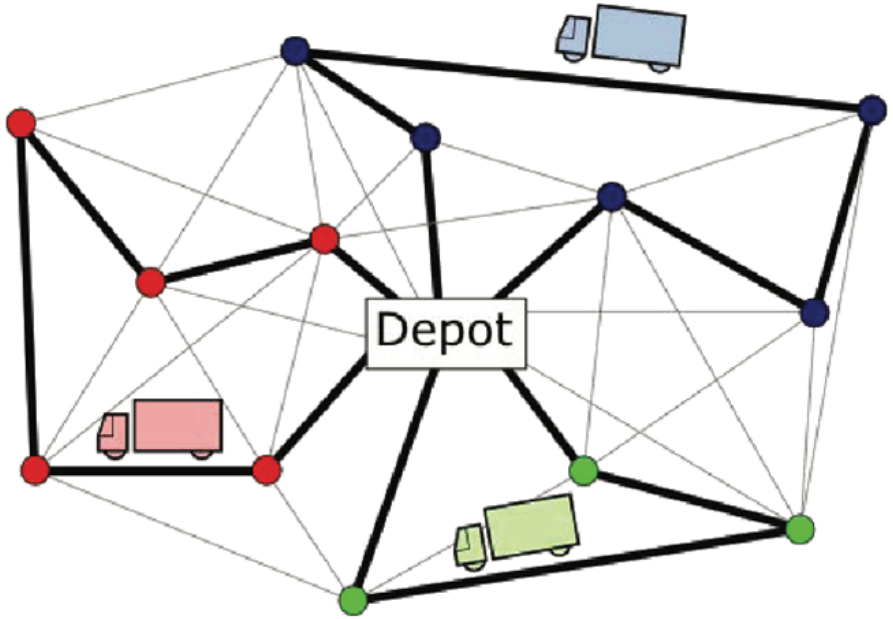


Figure: Vehicle routing problem

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:

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

- 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:

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

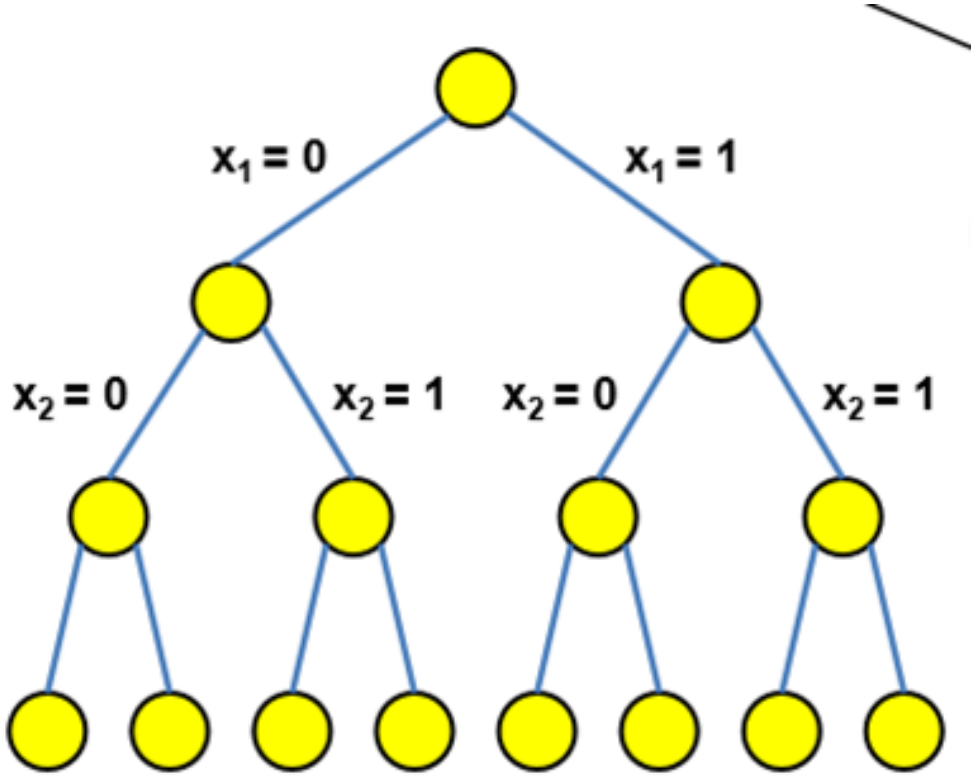
where $b = (b_1, \ldots, b_n)^T$ and U_b is the upper borrowing limit from the risk-free asset.

- Integer programming model:

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

Branch and bound-enumeration

- Enumeration (exponentially possible points)
- Example: knapsack problem



Branch and bound- LP relaxation

- Can we do it much cleverer?
- Linear programming relaxation

$$(LP)\max \;\; c^{\mathrm{T}}x$$
$$\text{s.t.} \;\; Ax = b \quad x \geq 0 \quad x \in Z^n.$$

LP relaxation

$$(LP)\max \;\; c^{\mathrm{T}}x$$
$$\text{s.t.} \;\; Ax = b \quad x \geq 0.$$

Why is LP relaxation useful?

- Suppose x* is the optimal solution of LP relaxation. If x* is integer, then x* solves IP.
- If x* is not integer, we create two subproblems.

  Let z* be the feasible value.

  Subproblem will be discarded if

  1) Linear relaxation is not feasible.

  2) The optimal objective of linear relaxation is less then z^*.

Branch and Bound Strategy (1)

Step1: Solve the linear relaxation of the problem. If solution is integer, then we are done. Otherwise, find a feasible objective value z^* and mark the direct descendants of this node as active, and go to step 2.

Step2: While there remain active nodes Select an active node j and mark it as inactive. Let $x(j)$ and $z(j)$ denote the optimal solution and objective of the LP relaxation of problem (j).

Branch and Bound Strategy (2)

Case 1: If $z^* >= z(j)$ then delete node j.

Case 2: If $z^* < z(j)$ and $x(j)$ is feasible for IP then update $z^* := z(j)$ and replace the incumbent by $x(j)$. delete node j.

Case 3: If $z^* < z(j)$ and $x(j)$ is not feasible for IP then mark the direct descendants of node j as active. End while.

How to create subproblems?

How to Branch?

- We want to divide the current problem into two or more subproblems that are easier than the original. A commonly used branching method:

$$x_i \leq \lfloor x_i^* \rfloor, \; x_i \geq \lceil x_i^* \rceil,$$

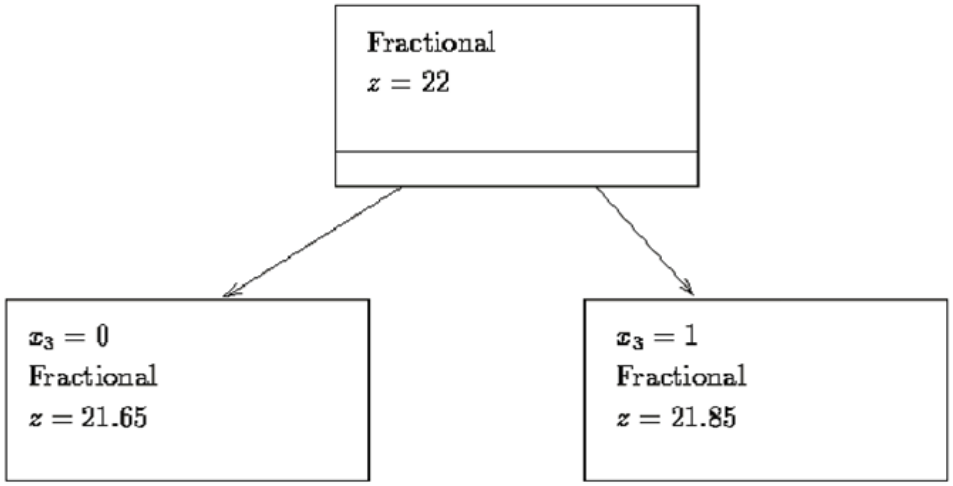
  where x_i^* is a fractional variable.
- Which variable to branch? A commonly used branching rule: Branch the most fractional variable.
- We would like to choose the branching that minimizes the sum of the solution times of all the created subproblems.
- How do we know how long it will take to solve each subproblem?
  Answer: We don't.
  Idea: Try to predict the difficulty of a subproblem.

- Example. Consider the following 0-1 knapsack problem:

$$\begin{aligned} \max \ & 8x_1 + 11x_2 + 6x_3 + 4x_4 \\ \text{s.t.} \ & 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14 \\ & x \in \{0,1\}^4. \end{aligned}$$

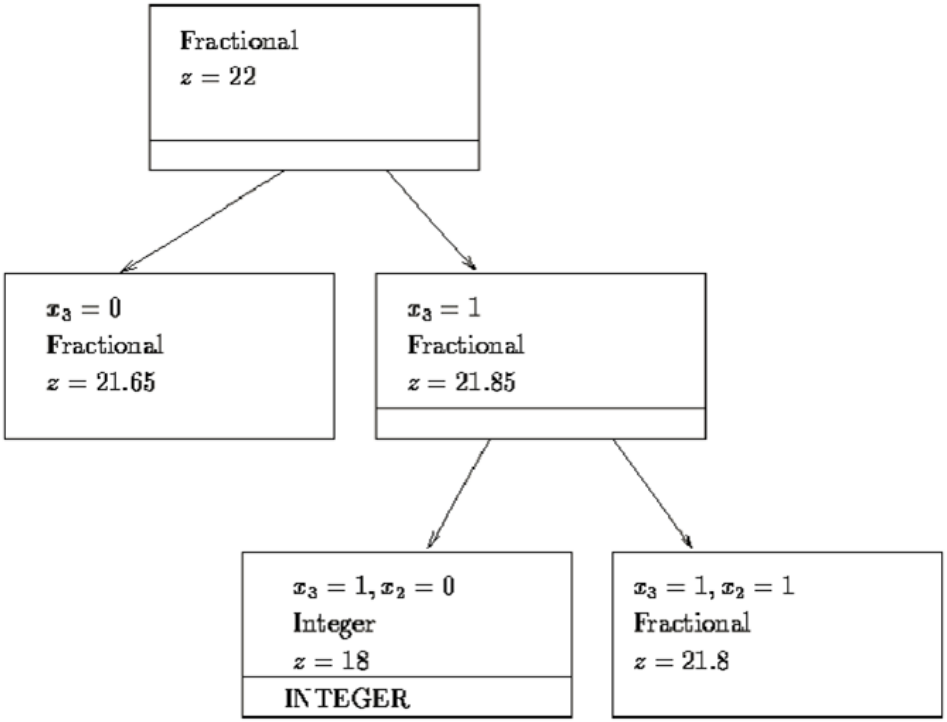
- The linear relaxation solution is $x = (1, 1, 0.5, 0)$ with a value of 22. The solution is not integral.
- Choose x_3 to branch. The next two subproblems will have $x_3 = 0$ and $x_3 = 1$ respectively.

- The search tree is:



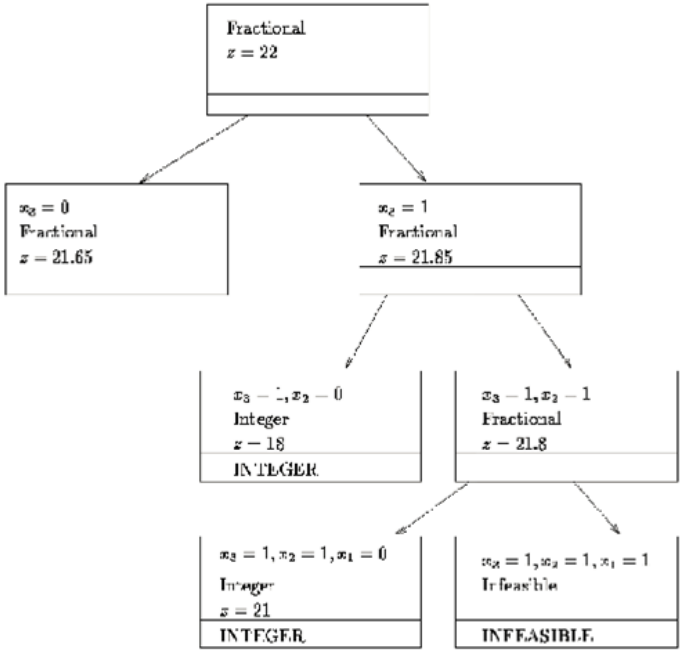
- The linear relaxation solutions to the two subproblems are
  - $x_3 = 0$: $x = (1, 1, 0, 0.667)$, objective value=21.65;
  - $x_3 = 1$: $x = (1, 0.714, 1, 0)$, objective value=21.85.
- At this point we know that the optimal value is no more than 21.85 (we actually know it is less than or equal to 21. but we still do not have any feasible integer solution. So, we will take a subproblem and branch on one of its variables.

- Choose the node with $x_3 = 1$ and branch on x_2. The branch tree is:



- The linear relaxation solutions are
  - $x_3 = 1$, $x_2 = 0$: $x = (1, 1, 1, 1)$, objective value=18;
  - $x_3 = 1$: $x_2 = 1$: $x = (0.6, 1, 1, 0)$, objective value=21.8.

- Choose node with $x_3 = 1$, $x_2 = 0$ and branch on x_1. The search tree is



- The linear relaxation solutions are
  - $x_3 = 1$, $x_2 = 0$, $x_1 = 0$: $x = (0, 1, 1, 1)$, objective value=21;
  - $x_3 = 1$: $x_2 = 1$: $x_1 = 1$: infeasible.
- The optimal solution is $x = (0, 1, 1, 1)$.

- Theorem (Dantzig (1957)): *The optimal solution to the continuous relaxation of* $(01KP)$ *is*

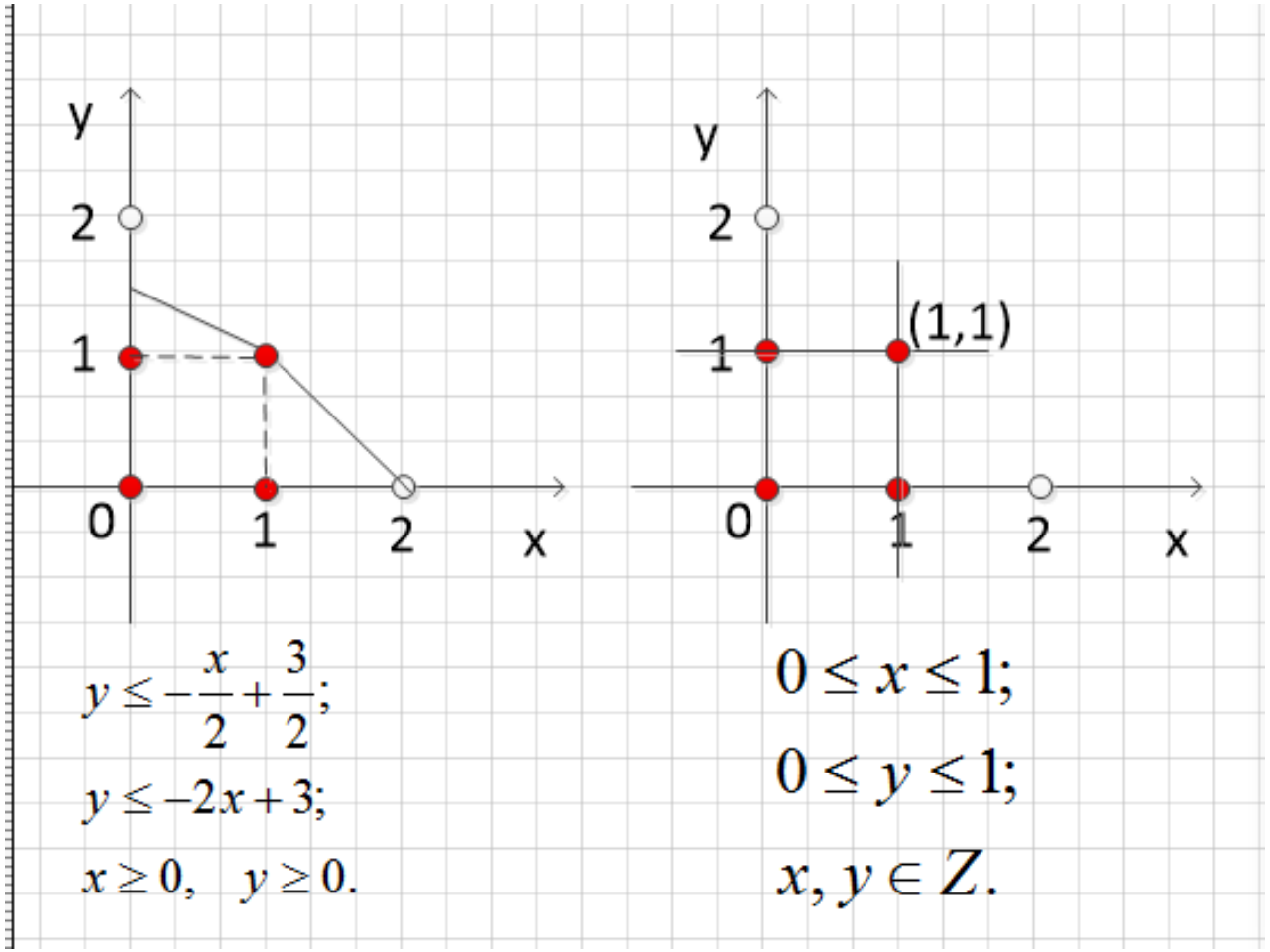
$$
\begin{aligned}
w_j &= 1,\ j = 1, \ldots, s, \\
w_j &= 0,\ j = s+2, \ldots, N, \\
w_{s+1} &= (b - \sum_{j=1}^{s} a_j)/a_{s+1}.
\end{aligned}
$$

- If c_j, $j = 1, \ldots, N$, are positive integers, then an upper bound of the optimal value of (LKP) is given by

$$
UB = \sum_{j=1}^{s} c_j + \lfloor (b - \sum_{j=1}^{s} a_j) c_{s+1}/a_{s+1} \rfloor, \tag{3}
$$

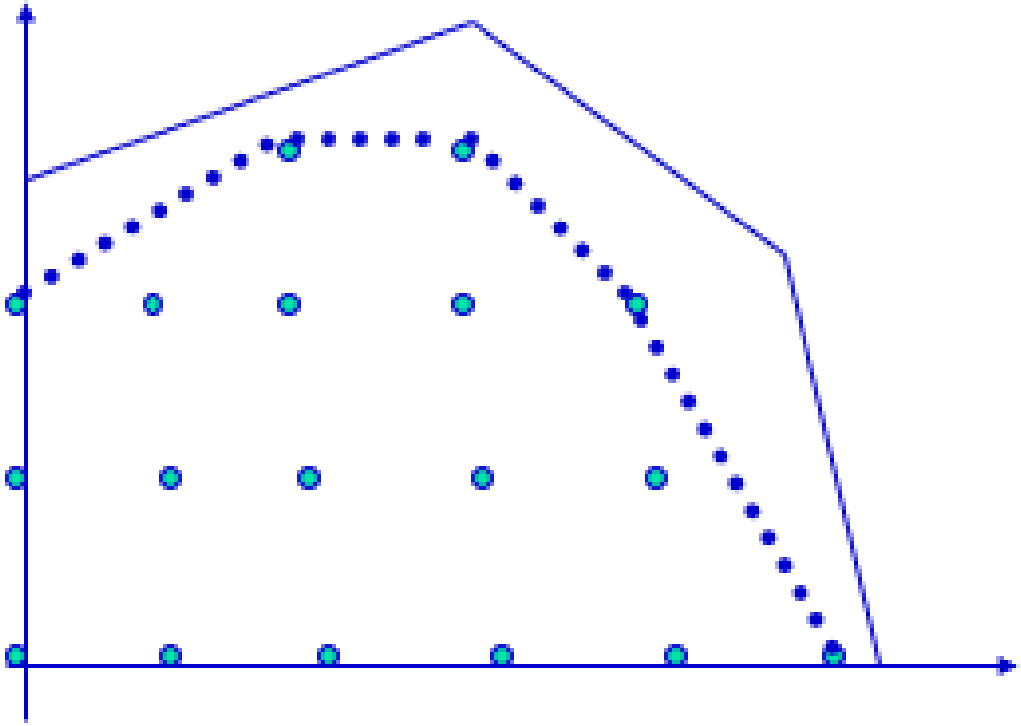
Cutting Plane (1)

One motivation



Cutting Plane (2)

Essential idea: step by step, go to the convex hull of feasible region



How to cut? (1)

Gomory's Cutting Plane Method

Ralph E. Gomory (1929–)

"Outline of an Algorithm for Integer Solutions to Linear Programs"
Bulletin of the American Mathematical Society, Vol. 64, pp. 275-278, 1958

How to cut? (2)

- Gomory's original idea: objective rounding

  May not work, non-integer solution may generate integer objective value.

- Advanced techniques: constraint rounding use simplex method

How to cut? (3)

- Cut the optimal solution of LP relaxation, keep the feasible integer solutions

x_B	x_N	
I	$B^{-1}N$	$B^{-1}b \geq 0$
$\sigma_B = 0$	σ_N	$c_B^T B^{-1} b$

$$x_i + \sum_{j \in N} \bar{a}_{ij} x_j = \bar{b}_i, \ i \in B$$

$$x_j^* = \begin{cases} \bar{b}_j & j \in B \\ 0 & j \in N \end{cases}$$

若 $\bar{b}_r$ 不是整数 $\longrightarrow x_r + \sum_{j \in N} \bar{a}_{rj} x_j = \bar{b}_r$

How to cut? (4)

$$x_r + \sum_{j \in N} \bar{a}_{rj} x_j = \bar{b}_r$$

$$\bar{a}_{rj} = \lfloor \bar{a}_{rj} \rfloor + f_{rj} \qquad \bar{b}_r = \lfloor \bar{b}_r \rfloor + f_r$$

$$0 \le f_{rj} < 1 \quad \bar{a}_{rj} \ge \lfloor \bar{a}_{rj} \rfloor \qquad \bar{b}_r > \lfloor \bar{b}_r \rfloor \quad 0 < f_r < 1$$

$$x_r + \sum_{j \in N} \lfloor \bar{a}_{rj} \rfloor x_j - \lfloor \bar{b}_r \rfloor = \left(f_r - \sum_{j \in N} f_{rj} x_j \right)$$

Integer feasible solution

Optimal basis feasible solution

$$x_r + \sum_{j \in N} \lfloor \bar{a}_{rj} \rfloor x_j \le \lfloor \bar{b}_r \rfloor \qquad x_r + \sum_{j \in N} \lfloor \bar{a}_{rj} \rfloor x_j > \lfloor \bar{b}_r \rfloor$$

How to cut? (5)

$$\min c^{\mathrm{T}} x$$
$$s.t.\begin{cases} Ax = b \\ x \geq 0, x \text{ is integer} \end{cases}$$

$$\min c^{\mathrm{T}} x$$
$$s.t.\begin{cases} Ax = b \\ x_r + \sum_{j \in N} \lfloor \bar{a}_{rj} \rfloor x_{rj} \leq \lfloor \bar{b}_r \rfloor \\ x \geq 0, x \text{ is integer} \end{cases}$$

$$\min c^{\mathrm{T}} x$$
$$s.t.\begin{cases} Ax = b \\ x_r + \sum_{j \in N} \lfloor \bar{a}_{rj} \rfloor x_{rj} + s_r = \lfloor \bar{b}_r \rfloor \\ x \geq 0, x \text{ is integer} \end{cases}$$

Example (1)

Gomory Cut Example: Start with the integer program

$$
\begin{aligned}
\min -x_1 - x_2 & \\
x_1 - x_2 + x_3 &= 2 \\
4x_1 + 9x_2 + x_4 &= 18 \\
-2x_1 + 4x_2 + x_5 &= 4 \\
x_1, x_2 &\geq 0
\end{aligned}
$$

Example (2)

-0.00	-0.00	-0.38	-0.15	-0.00	-3.54
1.00	0.00	0.69	0.08	0.00	2.77
0.00	1.00	-0.31	0.08	0.00	0.77
0.00	0.00	2.62	-0.15	1.00	6.46

Generate a cut based on the fractional variable $x_2 = 0.77$

$$x_2 - .31x_3 + 0.08x_4 = 0.77$$

Round up: $x2x3 <= 0$

Software for integer programming

- Matlab code: `bintprog` for solve binary integer linear programming problems.
- Commercial optimization software `CPLEX` and `Gurobi` for solving mixed integer linear and quadratic programming problems.
- Two simple ways of calling CPLEX MIP solver:
  - Optimization Programming Language (OPL) in CPLEX Optimization Studio.
  - CPLEX Matlab interface.
- Examples using Matlab and CPLEX

Branch-and-Cut Method

- Branch and cut is an LP-based branch and bound scheme in which the linear programming relaxations combined with by cutting plane method.
- The valid inequalities are generated dynamically using separation procedures.
- At each node of the search tree, cuts are generated and used to improve the LP relaxation.
- Branch-and-cut method is very efficient for some hard integer programming problems.

Background of the problem

In 2006, ICBC had more than 16,000 domestic branches, over 200 million individual customers, and more than 3.6 million corporate customers.

In the 1990s, large-scale expansion, one-sided pursuit of large, wide, and full, lack of foresight, unscientific site selection, resulting in low efficiency and high cost of network operations.

Rapid changes in China's economic level, urbanization, modernization, and so on $\cdots$

Such as the emergence of new urban areas, satellite cities $\cdots$

Figure 1: The figure shows ICBC branches and geographic entities (e.g., bank branches designated by circled I, office buildings designated by gray dots, and residential buildings designated by black dots) in a typical city in China.

Quantitative Methods in MS

Machine Learning: From Data to Prediction

(Data processing)

Operations Research: From Model to Decision

Location Problems and Models (1)

The problem: A city is divided into several regions, which are recorded as a set J; a sub-set I is selected from J, as a candidate site for establishing a bank network, and the number of outlets does not exceed N.

Decision variables: whether to establish a network point in $i \in I$ or not;

$$x_1 = \begin{cases} 1, & \text{Yes;} \\ 0, & \text{No.} \end{cases}$$

Constraint: the number of dots does not exceed N

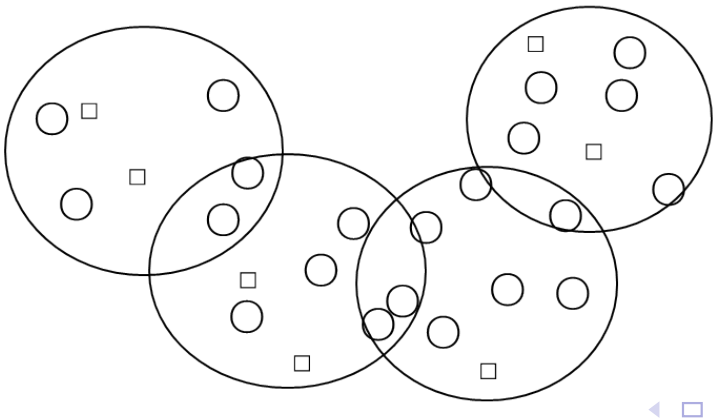
$$\sum_{i \in I} x_i \leq N.$$

Location Problems and Models (2)

Objection: The market potential coverage is the largest

$$\begin{aligned} \max \quad & \sum_{j \in J} f_j P_j \left(x_1, x_2 \to x_{|I} \right) \\ \text{s.t.} \quad & x_i \in \{0,1\}, \quad \forall i \in I, \\ & \sum_{i \in I} x_i \leq N, \end{aligned}$$

where x_i is whether to establish a network point in $i \in I$ or not, f_j is the market potential in j, P_j is the rate of coverage by corporate banking outlets in j.



ICBC Branch Reconfiguration Practice

Partials a city into a grid of tens of thousands of cells

Cell: 100 meters by 100 meters

Suzhou: 400 square kilometers; 400,000 cells

China: 300 cities; 12million cells

Data available: a list of metrics for GIS data
for example: total households of residential buildings, the number of supermarkets, companies

Cells

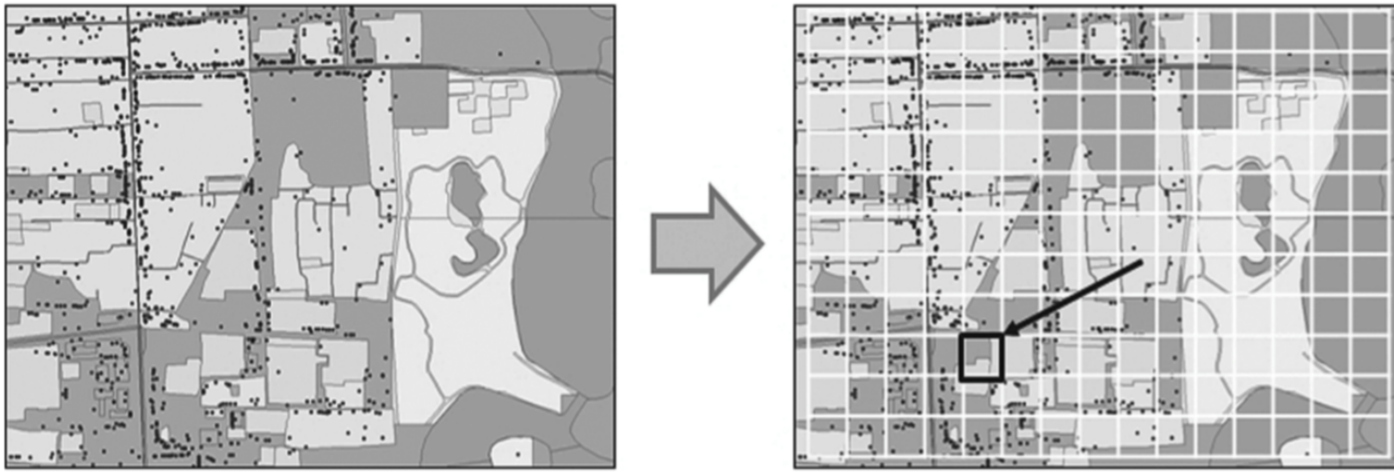


Figure A.1: BR divides a city area (left side of figure) into cells of 100 meters by 100 meters (right side of figure). The black box in the right figure represents such a cell.

GIS Data

Geographic entity category	Demographic attributes
Office building	Area size, rent price
Residential building	Area size, number of households
Shopping	Area size
Education	Area size, number of students
Entertainment	Area size
...	...

Table A.1: The table shows an example of a GIS data content list.

Market Potential Prediction Model (1)

The number of cells is huge; requires a highly efficient to predict the market potential

Linear model:

$$f = w^T x$$

where x is a vector of the metrics of the cell, w is a vector of weights.

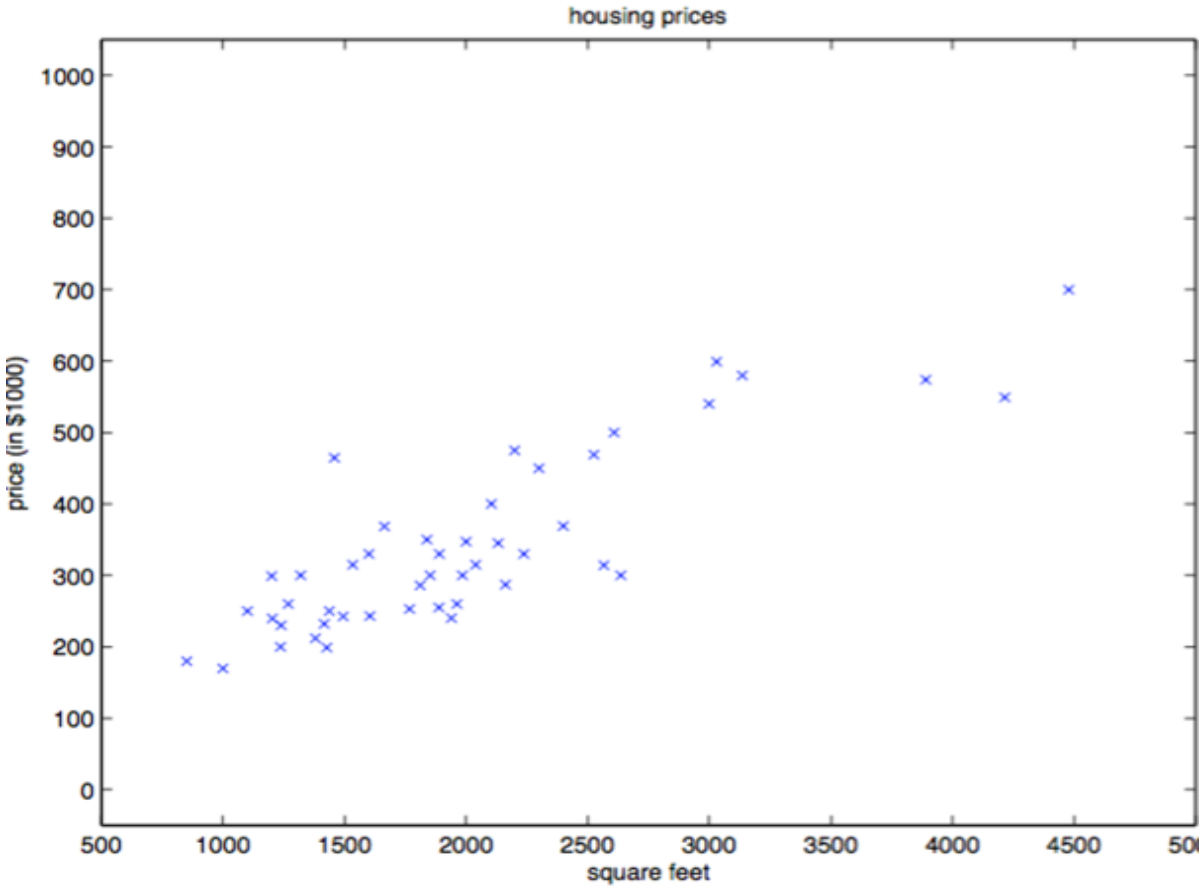
Question: estimate the weight w?

where x is a vector of the metrics of the cell, w is a vector of weights.

Linear Regression (1)

Living area (feet^2)	Price (1000\$s)
2104	400
1600	330
2400	369
1416	232
3000	540
$\vdots$	$\vdots$

Given data like this, how can we learn to predict the prices of other houses in Portland, as a function of the size of their living areas?


housing prices
price (in $1000)
square feet
1000
900
800
700
600
500
400
300
200
100
0
500
1000
1500
2000
2500
3000
3500
4000
4500

Linear Regression (2)

Linear Mapping

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x$$

Given a set of sample (x_i, y_i), the error Loss function is defined as

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} \left(h_\theta \left(x^{(i)} \right) - y^{(i)} \right)^2$$

Linear Regression (3)

Estimate the weight w by solving the optimization problem:

$$\min_{\theta} J(\theta) = \frac{1}{2} \sum_{i=1}^{m} \left(h_\theta \left(x^{(i)} \right) - y^{(i)} \right)^2$$

Market Potential Prediction Model (1)

A solution: given a set of cells xi, let the experts score the market potential; use the linear regression model to estimate the weight w.

Can we trust the experts?

Market Potential Prediction Model (2)

Cell preference: Given a pair of cells, experts can know which cell has more market potential,

$$w^T x_i - w^T x_j \geq 0$$

Metric preference: given a pair of metrics, experts can know which metric is more important.

$$w_m \geq w_n, \quad w_m \geq 0, \quad w_n \geq 0$$

Market Potential Prediction Model (4)

The model:

$$\min_{w} \quad \left\{ \lambda_1 \|w\|_2^2 + \lambda_2 f^T L f + \lambda_3 \sum_{ij} \xi_{ij} + \lambda_4 \sum_{ij} \xi_{ij}^* \right\},$$

$$\text{s.t. } \forall (x_i, x_j) \in E_{ss}, \quad w^T x_i - w^T x_j \geq 1 - \xi_{ij},$$

$$\forall (w_i, w_j) \in E_m, \quad w_j - w_i \leq \xi_{ij}^*,$$

$$w \geq 0 \quad \xi_{ij} \geq 0, \quad \xi_{ij}^* \geq 0.$$

objective: ensure that the predicted values of two similar samples are close.

Branch Network Optimization Model (1)

Problem Setting:

1) N branches are to located in J candidate locations;
2) I is the set of all cells;
3) Each branch can exactly one of K formats;
4) N_k is the maximum number of k-type branches;
5) S_k is the maximum coverage distance of a kth-type branch

Branch Network Optimization Model (2)

Decision variable:

$$x_j^k \in \{0,1\}, \quad \forall j \in J, k \in K$$

$x_j^k = 1$ if a k-type branch is located at candidate location j; otherwise, $x_i^k = 0$

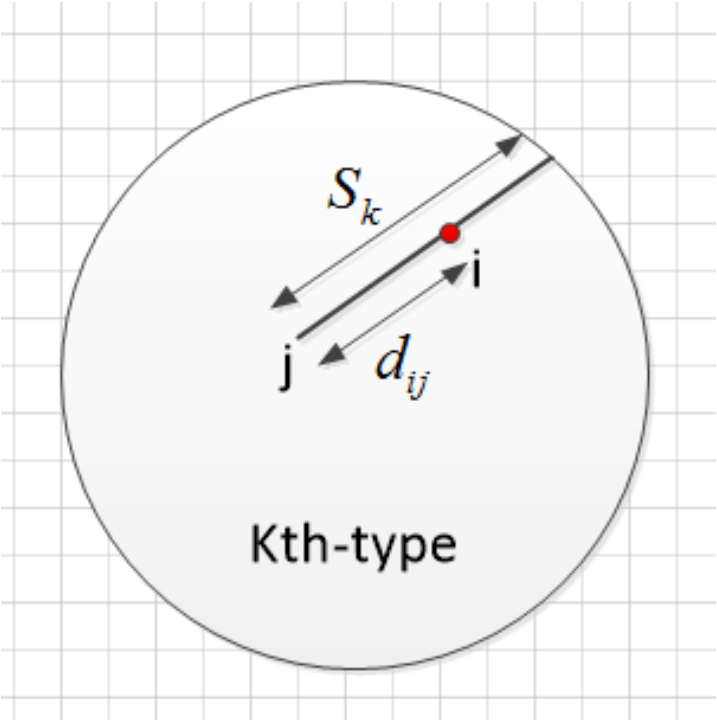
Constraints:

$$\sum_{k \in K} x_j^k \leq 1, \quad \forall j \in J,$$

$$\sum_{k \in K} x_j^k \leq N^k, \quad \forall k \in K.$$

Branch Network Optimization Model (3)

Covered percentage of cell i by a kth-type brach at site j:



$$f^k\left(d_{ij}\right) = \max\left\{1 - \left(d_{ij}/S^k\right), 0\right\}$$

Branch Network Optimization Model (4)

The total covered percentage of cell i:

$$y_i = \max\left\{1, \sum_{j\in J}\sum_{k\in K} f^k\left(d_{ij}\right) x_j^k\right\}, \quad \forall i \in I$$

The objective:

$$\max\left\{\sum_{i\in I} w_i y_i\right\}$$

w_i is the market potential value in ith cell.

Branch Network Optimization Model (5)

The model:

$$\max \left\{ \sum_{i \in I} w_i y_i \right\}$$

$$\text{s.t.} \quad \sum_{k \in K} x_j^k \leq 1, \quad \forall j \in J$$

$$\sum_{k \in K} x_j^k \leq N^k, \quad \forall k \in K$$

or equivalent:

$$\max \left\{ \sum_{i \in I} w_i y_i \right\}$$

$$\text{s.t.} \quad \sum_{k \in K} x_j^k \leq 1, \quad \forall j \in J,$$

$$\sum_{k \in K} x_j^k \leq N^k, \quad \forall k \in K,$$

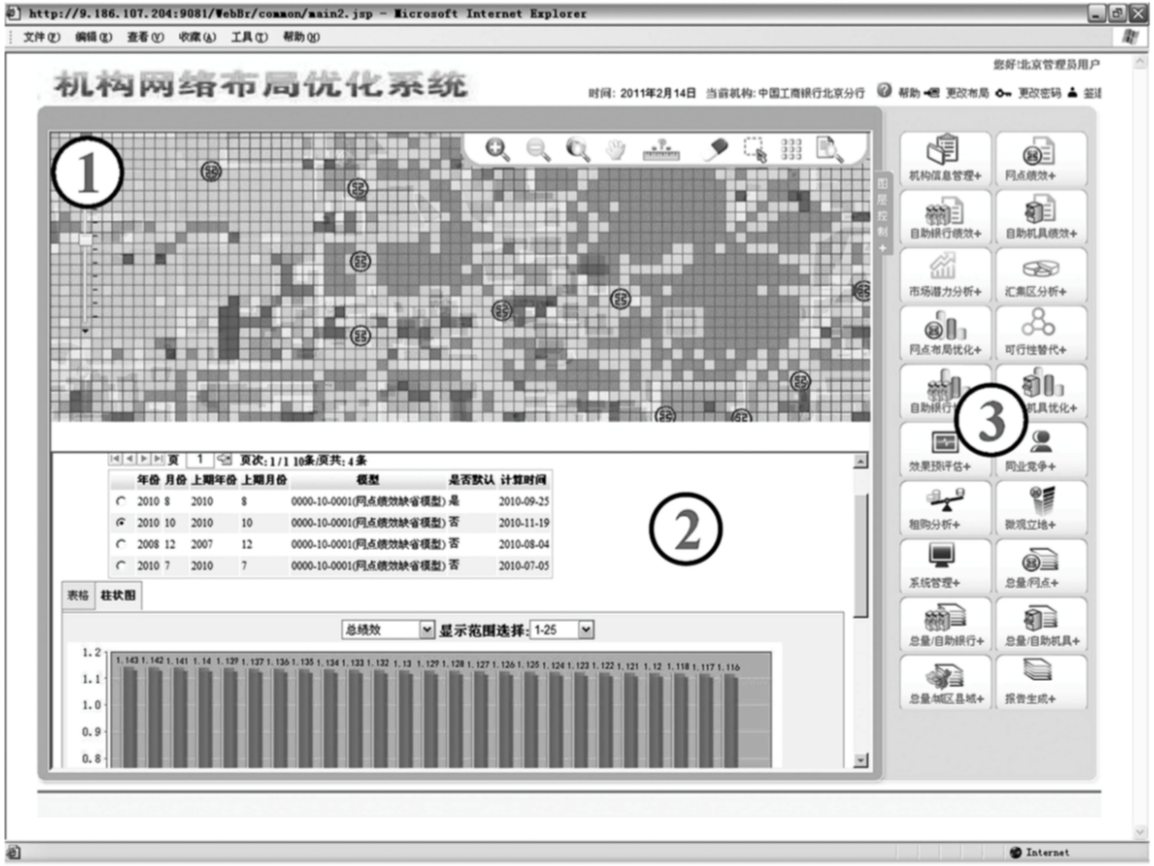
$$y_i \leq 1, \quad \forall i \in I,$$

$$y_i \leq \sum_{j \in J} \sum_{k \in K} f^k \left(d_{ij} \right) x_j^k, \quad \forall i \in I$$

Branch Network Optimization Model (6)

The Challenge: typically, K=3, I =40,000, J =4000 (midsize city); Beijing and Shanghai, I =120,000, J= 12,000

Large-scale optimization problems; needs efficient algorithm; Bender decomposition


http://9.186.107.204:9081/WebBr/common/main2.jsp - Microsoft Internet Explorer
机构网络布局优化系统
1
2
3
Internet

Practice in Suzhou

ICBC' s ability to reach new individual and small-to-medium commercial customers has been enhanced greatly

Branch performance has significantly improved

The increase in deposits was US 1.04 billion;
such deposits are expected to increase by ten of billions of US dollars