Outline

1. Integer Programming

2. Modeling
   - Assignment Problem
   - Knapsack Problem
   - Set Covering
   - Graph Problems
A precise analysis of running time for an algorithm includes the number of bit operations together with number of arithmetic operations.

**Strongly polynomial algorithms**: the running time of the algorithm is independent on the number of bit operations. Eg: same running time for input numbers with 10 bits as for inputs with a million bits.

No strongly polynomial-time algorithm for LP is known.

Running time depends on the sizes of numbers. We have to restrict attention to rational instances when analyzing the running time of algorithms and assume they are coded in binary.

**Theorem**

*Optimal feasible solutions to LP problems are always rational as long as all coefficient and constants are rational.*

Proof: derives from the fact that in the simplex we only perform multiplications, divisions and sums of rational numbers.
Outline

1. Integer Programming

2. Modeling
   Assignment Problem
   Knapsack Problem
   Set Covering
   Graph Problems
Discrete Optimization

- Often need to deal with integral inseparable quantities.
- Sometimes rounding can go.
- Other times rounding not feasible: eg, presence of a bus on a line is 0.3...
**Integer Linear Programming**

The world is not linear: OR is the art and science of obtaining bad answers to questions to which otherwise worse answers would be given.

\[
\begin{align*}
\text{max } & \quad c^T x \\
\text{max } & \quad c^T x \\
& \quad Ax \leq b \\
& \quad x \geq 0 \\
& \quad x \text{ integer}
\end{align*}
\]

\[
\begin{align*}
\text{max } & \quad c^T x + h^T y \\
\text{max } & \quad c^T x \\
& \quad Ax + Gy \leq b \\
& \quad x \geq 0 \\
& \quad y \geq 0 \\
& \quad y \text{ integer}
\end{align*}
\]

Linear Programming (LP)

Integer (Linear) Programming (ILP)

Binary Integer Program (BIP)

Mixed Integer Programming (MILP)

\[
\begin{align*}
\text{max } & \quad f(x) \\
& \quad g(x) \leq b \\
& \quad x \geq 0
\end{align*}
\]

Non-linear Programming (NLP)
Recall:

- $\mathbb{Z}$ set of integers
- $\mathbb{Z}^+$ set of positive integer
- $\mathbb{Z}_0^+$ set of nonnegative integers ($\{0\} \cup \mathbb{Z}^+$)
Definition (Combinatorial Optimization Problem (COP))

Given: Finite set $N = \{1, \ldots, n\}$ of objects, weights $c_j \forall j \in N$, $\mathcal{F}$ a collection of feasible subsets of $N$

Find a minimum weight feasible subset:

$$\min_{S \subseteq N} \left\{ \sum_{j \in S} c_j \mid S \in \mathcal{F} \right\}$$

Many COP can be modelled as IP or BIP.

Typically: incidence vector of $S$, $x^S \in \mathbb{R}^n$: $x_j^S = \begin{cases} 1 & \text{if } j \in S \\ 0 & \text{otherwise} \end{cases}$
Rounding

\[
\begin{align*}
\text{max } & 100x_1 + 64x_2 \\
\text{s.t. } & 50x_1 + 31x_2 \leq 250 \\
& 3x_1 - 2x_2 \geq -4 \\
& x_1, x_2 \in \mathbb{Z}^+ \\
\end{align*}
\]

LP optimum \((376/193, 950/193)\)
IP optimum \((5, 0)\)

\(\leadsto\) feasible region convex but not continuous: Now the optimum can be on the border (vertices) but also internal.
Possible way: solve the relaxed problem.

- If solution is integer, done.
- If solution is rational (never irrational) try rounding to the nearest integers (but may exit feasibility region)
  - if in \(\mathbb{R}^2\) then \(2^2\) possible roundings (up or down)
  - if in \(\mathbb{R}^n\) then \(2^n\) possible roundings (up or down)

Note: rounding does not help in the example above
Cutting Planes

\[
\begin{align*}
\text{max } & \quad x_1 + 4x_2 \\
& \quad x_1 + 6x_2 \leq 18 \\
& \quad x_1 \leq 3 \\
& \quad x_1, x_2 \geq 0 \\
& \quad x_1, x_2 \text{ integer}
\end{align*}
\]
Branch and Bound

max \ x_1 + 2x_2 \\
\ x_1 + 4x_2 \leq 8 \\
4x_1 + x_2 \leq 8 \\
x_1, x_2 \geq 0, \text{integer}
Integer Programming
Modeling

4.8

\[ x_1 \leq 1 \]

\[ x_1 \geq 2 \]

\[ 4x_1 + x_2 = 8 \]

\[ x_1 + 4x_2 = 8 \]

\[ x_1 + 2x_2 = 1 \]

\[ 4x_1 + x_2 = 8 \]

\[ x_1 + 2x_2 = 1 \]

\[ 4x_1 + x_2 = 8 \]
Integer Programming
Modeling

\[
\begin{align*}
4.8 - \infty & \quad x_2 \leq 1 \\
4.5 - \infty & \quad x_2 \leq 1 \\
5 & \quad x_1 = 2, \quad x_2 = 0 \\
3 & \quad x_1 = 1, \quad x_2 = 1 \\
4 & \quad x_1 = 0, \quad x_2 = 2 \\
-\infty & \quad x_1 \geq 2 \\
4 & \quad x_2 \geq 2
\end{align*}
\]

\[
\begin{align*}
x_1 + 4x_2 &= 8 \\
x_1 + 2x_2 &= 1 \\
4x_1 + x_2 &= 8
\end{align*}
\]
Outline

1. Integer Programming

2. Modeling
   - Assignment Problem
   - Knapsack Problem
   - Set Covering
   - Graph Problems
Find out exactly what the decision makes needs to know:

- which investment?
- which product mix?
- which job \( j \) should a person \( i \) do?

Define Decision Variables of suitable type (continuous, integer valued, binary) corresponding to the needs

Formulate Objective Function computing the benefit/cost

Formulate mathematical Constraints indicating the interplay between the different variables.
How to “build” a constraint

- Formulate relationship between the variables in plain words
- Then formulate your sentences using logical connectives and, or, not, implies
- Finally convert the logical statement to a mathematical constraint.

Example

- “The power plant must not work in two neighbouring time periods”
- on/off is modelled using binary integer variables
- \( x_i = 1 \) or \( x_i = 0 \)
- \( x_i = 1 \) implies \( \Rightarrow x_{i+1} = 0 \)
- \( x_i + x_{i+1} \leq 1 \)
Outline

1. Integer Programming

2. Modeling
   Assignment Problem
   Knapsack Problem
   Set Covering
   Graph Problems
The Assignment Problem

The problem ..

The assignment problem is a well known optimization problem where assignees are being assigned to perform tasks. Assigning people to jobs is a common application of the assignment problem.

Suppose we have $n$ people and $n$ jobs, and that each person has a certain proficiency at each job.

Formulate a mathematical model that can be used to find an assignment that maximizes the total proficiency.
The Assignment Problem

Decision Variables:

\[ x_{ij} = \begin{cases} 1 & \text{if person } i \text{ is assigned job } j \\ 0 & \text{otherwise,} \end{cases} \text{ for } i, j = 1, 2, \ldots, n \]

Objective Function:

\[ \max \sum_{i=1}^{n} \sum_{j=1}^{n} \rho_{ij} x_{ij} \]

where \( \rho_{ij} \) is person \( i \)'s proficiency at job \( j \)
The Assignment Problem

**Constraints:**
Each person is assigned one job:

$$\sum_{j=1}^{n} x_{ij} = 1 \text{ for all } i$$

e.g. for person 1 we get $x_{11} + x_{12} + x_{13} + \cdots + x_{1n} = 1$

Each job is assigned to one person:

$$\sum_{i=1}^{n} x_{ij} = 1 \text{ for all } j$$

e.g. for job 1 we get $x_{11} + x_{21} + x_{31} + \cdots + x_{n1} = 1$
Outline

1. Integer Programming

2. Modeling
   Assignment Problem
   Knapsack Problem
   Set Covering
   Graph Problems
The Knapsack Problem

The problem ..

Given a set of \( n \) items, each with a value \( v_i \) and weight \( w_i \) \( (i = 1, \ldots, n) \), one must determine the number of each item to include in a collection so that the total weight is less than a given limit, \( W \), and the total value is as large as possible.

The “knapsack” name derives from the problem faced by someone who is constrained by a fixed-size knapsack and must fill it with the most useful items.

Assuming we can take at most one of any item, that \( \sum_i w_i > W \), formulate a mathematical model to determine which items give the largest value.

Model used, eg, in capital budgeting, project selection, etc.
The Knapsack Problem

Decision Variables:

\[ x_i = \begin{cases} 
1 & \text{if item } i \text{ is taken} \\
0 & \text{otherwise,} 
\end{cases} \quad \text{for } i = 1, 2 \ldots, n \]

Objective Function:

\[ \max \sum_{i=1}^{n} v_i x_i \]

Constraints:

Knapsack capacity restriction:

\[ \sum_{i=1}^{n} w_i x_i \leq W \]
Outline

1. Integer Programming

2. Modeling
   - Assignment Problem
   - Knapsack Problem
   - Set Covering
   - Graph Problems
Set Covering

**Given:** a number of regions, a number of centers, regions that can be served in less than 8 minutes, cost of installing an emergency center.

**Task:** Where to install a set of emergency centers such that the total cost is minimized and all regions safely served?

**As a COP:** \( M = \{1, \ldots, m\} \) regions, \( N\{1, \ldots, n\} \) centers, \( S_j \subseteq M \) regions serviced by \( j \)

\[
\min_{T \in N} \left\{ \sum_{j \in T} c_j \mid \bigcup_{j \in T} S_j = M \right\}
\]
As a BIP:

Variables:
\( x \in \mathbb{B}^n, \ x_j = 1 \) if center \( j \) is selected, \( 0 \) otherwise

Objective:
\[
\min \sum_{j=1}^{n} c_j x_j
\]

Constraints:

- incidence matrix: \( a_{ij} = \begin{cases} 1 \\ 0 \end{cases} \)

- \( \sum_{j=1}^{n} a_{ij} x_j \geq 1 \)
Example

- \( M = \{1, \ldots, 5\} \), \( N = \{1, \ldots, 6\} \), \( c_j = 1 \ \forall j = 1, \ldots, 6 \)
  \( S_1 = (1, 2) \), \( S_2 = (1, 3, 5) \), \( S_3 = (2, 4, 5) \), \( S_4 = (3) \), \( S_5 = (1) \), \( S_6 = (4, 5) \)

- 

\[
A = \begin{bmatrix}
1 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 1
\end{bmatrix}
\]
Set covering
cover each of M at least once
1. min, \( \geq \)
2. all RHS terms are 1
3. all matrix elements are 1

\[
\min c^T x \\
Ax \geq 1 \\
x \in \mathbb{B}^n
\]

Set packing
cover as many of M without overlap
1. max, \( \leq \)
2. all RHS terms are 1
3. all matrix elements are 1

\[
\max c^T x \\
Ax \leq 1 \\
x \in \mathbb{B}^n
\]

Set partitioning
cover exactly once each element of \( M \)
1. max or min, \( = \)
2. all RHS terms are 1
3. all matrix elements are 1

\[
\max c^T x \\
Ax = 1 \\
x \in \mathbb{B}^n
\]

Generalization: \( RHS \geq 1 \)
Application examples:

- Aircrew scheduling: \( M \): legs to cover, \( N \): rosters
- Vehicle routing: \( M \): customers, \( N \): routes
Manpower Planning

- Each person covers 7 hours
- A person starting in hour 3 contributes to the workload in hours 3, 4, 5, 6, 7, 8, 9
- A person starting in hour $i$ contributes to the workload in hours $i, \ldots, i + 6$

Modelling task
Formulate a mathematical model to determine the number of people required to cover the workload
Decision Variables:

- $x_i \in \mathbb{N}_0$: number of people starting work in hour $i (i = 1, \ldots, 15)$

Objective Function:

$$\min \sum_{i=1}^{9} x_i$$

Constraints:

- Demand:
  $$\sum_{i=t-6}^{i=t} x_i \geq d_t \text{ for } t = 1, \ldots, 15$$

- Bounds:
  $$x_{-5}, \ldots, x_0 = 0$$
A good written example:

2.1. Notation
Let $N$ be the set of operational flight legs and $K$ the set of aircraft types. Denote by $n^k$ the number of available aircraft of type $k \in K$. Define $\Omega^k$, indexed by $p$, as the set of feasible schedules for aircraft of type $k \in K$ and let index $p = 0$ denote the empty schedule for an aircraft. Next associate with each schedule $p \in \Omega^k$ the value $c^k_p$ denoting the anticipated profit if this schedule is assigned to an aircraft of type $k \in K$ and $a^k_{sp}$ a binary constant equal to 1 if this schedule covers flight leg $i \in N$ and 0 otherwise. Furthermore, let $S$ be the set of stations and $S^k \subseteq S$ the subset having the facilities to serve aircraft of type $k \in K$. Then, define $o^k_p$ and $d^k_p$ to equal to 1 if schedule $p$, $p \in \Omega^k$, starts and ends respectively at station $s$, $s \in S^k$, and 0 otherwise.

Denote by $\theta^k_p$, $p \in \Omega^k \setminus \{0\}$, $k \in K$, the binary decision variable which takes the value 1 if schedule $p$ is assigned to an aircraft of type $k$, and 0 otherwise. Finally, let $\theta^k_p$, $k \in K$, be a nonnegative integer variable which gives the number of unused aircraft of type $k$.

2.2. Formulation
Using these definitions, the DARSP can be formulated as:

$$\text{Maximize } \sum_{k \in K} \sum_{p \in \Omega^k} c^k_p \theta^k_p$$  \hspace{1cm} (1)

subject to:

$$\sum_{k \in K} \sum_{p \in \Omega^k} a^k_{sp} \theta^k_p = 1 \hspace{1cm} \forall i \in N,$$  \hspace{1cm} (2)

$$\sum_{p \in \Omega^k} (a^k_{sp} - o^k_{sp}) \theta^k_p = 0 \hspace{1cm} \forall k \in K, \forall s \in S^k,$$  \hspace{1cm} (3)

$$\sum_{p \in \Omega^k} \theta^k_p = n^k \hspace{1cm} \forall k \in K,$$  \hspace{1cm} (4)

$$\theta^k_p \geq 0 \hspace{1cm} \forall k \in K, \forall p \in \Omega^k,$$  \hspace{1cm} (5)

$$\theta^k_p \text{ integer } \forall k \in K, \forall p \in \Omega^k.$$  \hspace{1cm} (6)

The objective function (1) states that we wish to maximize the total anticipated profit. Constraints (2) require that each operational flight leg be covered exactly once. Constraints (3) correspond to the flow conservation constraints at the beginning and the end of the day at each station and for each aircraft type. Constraints (4) limit the number of aircraft of type $k \in K$ that can be used to the number available. Finally, constraints (5) and (6) state that the decision variables are nonnegative integers. This model is a Set Partitioning problem with additional constraints.

Outline

1. Integer Programming

2. Modeling
   - Assignment Problem
   - Knapsack Problem
   - Set Covering
   - Graph Problems
Matching

Definition (Matching Theory Terminology)

Matching: set of pairwise non adjacent edges
Covered (vertex): a vertex is covered by a matching $M$ if it is incident to an
to an edge in $M$
Perfect (matching): if $M$ covers each vertex in $G$
Maximum (matching): if $M$ covers as many vertices as possible
Matchable (graph): if the graph $G$ has a perfect matching

$$\max \sum_{v \in V} w_v x_v$$
$$\sum_{e \in E: v \in e} x_e \leq 1 \quad \forall v \in V$$
$$x_e \in \{0, 1\} \quad \forall e \in E$$

Special case: bipartite matching $\equiv$ assignment problems
Vertex Cover

Select a subset $S \subseteq V$ such that each edge has at least one end vertex in $S$.

$$\min \sum_{v \in V} x_v$$

$$x_v + x_u \geq 1 \quad \forall u, v \in V, uv \in E$$

$$x_v \in \{0, 1\} \quad \forall v \in V$$

Approximation algorithm: set $S$ derived from the LP solution in this way:

$$S_{LP} = \{v \in V : x_v^* \geq 1/2\}$$

(it is a cover since $x_v^* + x_u^* \geq 1$ implies $x_v^* \geq 1/2$ or $x_u^* \geq 1/2$)

Proposition

The LP rounding approximation algorithm gives a 2-approximation:

$$|S_{LP}| \leq 2|S_{OPT}|$$ (at most as bad as twice the optimal solution)

Proof: Let $\bar{x}$ be opt to IP. Then $\sum x_v^* \leq \sum \bar{x}_v$.

$$|S_{LP}| = \sum_{v \in S_{LP}} 1 \leq \sum_{v \in V} 2x_v^* \quad \text{since } x_v^* \geq 1/2 \text{ for each } v \in S_{LP}$$

$$|S_{LP}| \leq 2 \sum_{v \in V} x_v^* \leq 2 \sum_{v \in V} \bar{x}_v = 2|S_{OPT}|$$
Maximum independent Set

Find the largest subset $S \subseteq V$ such that the induced graph has no edges

$$\text{max} \quad \sum_{v \in V} x_v$$

$$x_v + x_u \leq 1 \quad \forall u, v \in V, uv \in E$$

$$x_v = \{0, 1\} \quad \forall v \in V$$

Optimal sol of LP relaxation sets $x_v = 1/2$ for all variables and has value $|V|/2$.

What is the value of an optimal solution of a complete graph?

LP relaxation gives an $O(n)$-approximation (almost useless)