Reconnect '04
Introduction to Integer Programming

Cynthia Phillips, Sandia National Laboratories

Integer programming (IP)

Min $c^T x$
Subject to:

$Ax = b$
$\ell \leq x \leq u$

$x = (x_I, x_C)$

$x_I \in \mathbb{Z}^n$ (integer values)

$x_C \in \mathbb{Q}^n$ (rational values)

- Can also have inequalities in either direction (slack variables):
  
  $a_i^T x \leq b_i \Rightarrow a_i^T x + s_i = b_i \quad s_i \geq 0$

- If $x_I = \emptyset$ and $x_C = \emptyset$ then this is a mixed-integer program (MIP)

- Linear programming (LP) has no integrality constraints $x = \emptyset$ (in P)

- IP (easily) expresses any NP-complete problem

Terminology

In this context programming means making decisions. Leading terms say what kind:

- (Pure) Integer programming: all integer decisions
- Linear programming
- Quadratic programming: quadratic objective function
- Nonlinear programming: nonlinear constraints
- Stochastic programming: finite probability distribution of scenarios

Came from operations research (practical optimization discipline)

Computer programming (by someone) is required to solve these.

Decisions

The IPs I've encountered in practice involve either:

- Allocation of scarce resources
- Study of a natural system
  - Computational biology
  - Mathematics

Maybe during or after this course, you can add to the list.
**Integer Variables**

Use \( x_i \in \{0, 1\} \) (binary variables) to model:
- Yes/no decisions
- Disjunctions
- Logical conditions
- Piecewise linear functions (this not covered in this lecture)

**General Integer Variables**

Use general integer variables to choose a number of indivisible objects such as the number of planes to produce:
- Integer range should be small (e.g. 1-10)
- Computational tractability
- Larger ranges may be well approximated by rational variables (number of bags of potato chips to produce)

**Example: Binary Knapsack**

Given set of objects 1..\( n \)
total weight \( W \), item weight/size \( w_i \), value \( v_i \)

\[
x_i = \begin{cases} 
1 & \text{If we select item } i \\
0 & \text{Otherwise} 
\end{cases}
\]

\[
\max \sum w_i x_i 
\]

Subject to

\[
\sum w_i x_i \leq W 
\]

**Example: Shortest-Path Network Interdiction**

Delay an adversary moving through a network.
- Adversary moves start→target along a shortest path (in worst case)
- Path length = sum of edge lengths. Measure of time, exposure, etc.
Example: Shortest-Path Network Interdiction

Defender blocks the intruder by paying to increase edge lengths.
Goal: Maximize the resulting shortest path.

Graph G = (V, E)
Edge lengths \( f_{uv} \) for edge (u,v)
Can increase length of (u,v) by \( \lambda_{uv} \) at cost \( c_{uv} \)
Budget B

Variables:
- \( d_u \): shortest distance from start s to node u
- \( x_{uv} \): 1 if we pay to lengthen edge (u,v); 0 otherwise

Objective: maximize the shortest path to the target
maximize \( d_t \)

Subject to:
Path to the start has length 0:
\( d_s = 0 \)

Calculate a shortest path length:
\[
\begin{align*}
    d_u &= d_s + f_{su} + \lambda_{su} x_{su} \text{ for all } (u,v) \in E \\
    d_u &= d_u + f_{uv} + \lambda_{uv} x_{uv} \text{ for all } (u,v) \in E \\
\end{align*}
\]

Respect the budget:
\[
\sum_{u \neq s} f_{uv} x_{uv} \leq B
\]
Example: Unconstrained Facility Location

Given potential facility locations, $n$ customers to be served

$\ell_j$ = cost to build facility $j$

$h_{ij}$ = cost to meet all of customer $i$'s demand from facility $j$

Sometimes it's OK to satisfy customers from multiple facilities:

$x_j = \begin{cases} 
1 & \text{if facility } j \text{ built} \\
0 & \text{Otherwise}
\end{cases}$

$y_{ij} = \begin{cases} 
1 & \text{if customer } i \text{ is served by facility } j \\
0 & \text{Otherwise}
\end{cases}$

\[
\min \sum_j \ell_j x_j + \sum_{i,j} h_{ij} y_{ij}
\]

s.t. $y_i = \sum_j y_{ij}$ (each customer satisfied)

$y_i \leq x_j \quad \forall i,j$ (facility built before use)

$x_j, y_{ij} \in \{0,1\}$

Sometimes it's OK to satisfy customers from multiple facilities:

$y_{ij}$ becomes a percentage: $y_{ij} \in [0,1]$

Formulation is really important in practice

Unconstrained facility location

Could sum constraints $y_{i,j} \forall j$ over all customers $i$ to get:

\[
\sum_j y_{i,j} \leq n_i
\]

Recall $n$ is the number of customers.

Still requires a facility is built before use (IPs are equivalent at optimality)

But, for 40 customers, 40 facilities, random costs

• First formulation solves in 2 seconds
• Second formulation solves in 53,121 seconds (14.75 hours)

What makes one formulation so much better?

• Understanding this fully is an open problem.

• Some performance differences can be explained by the way IPs are solved in practice by branch-and-bound-like algorithms: the LP relaxation

Recall Integer Programming (IP)

\[
\min c^T x
\]

Subject to:

$Ax = b$

$\ell \leq x \leq u$

$x \in \mathbb{Z}^n$ (integer values)

$x \in \mathbb{Q}^n$ (rational values)
Linear programming (LP) relaxation of an IP

$$\text{Min} \quad c^T x$$

Subject to:

$\ell \leq x \leq u$

$x = (x_I, x_C)$

(integer values)

$x_C \in Q^n$ (rational values)

- LP can be solved efficiently (in theory and practice)
- Relaxation = removing constraints
  - All feasible IP solutions are feasible
  - LP gives a lower bound

Linear Programming Geometry

The solutions to a single inequality $a^T x \leq b$, $x \in Q^n$ form a half space (in n-dimensional space)

Linear Programming Geometry

Intersection of all the linear inequalities form a convex polytope

- For simplicity, we’ll always assume polytope is bounded

IP Geometry

Feasible integer points form a lattice inside the LP polytope
IP/Geometry

The convex hull of this lattice forms the integer polytope.

IP/LP Geometry

A "good" formulation keeps this region small.

Every node for which the LP bound is lower than the integer optimal must be processed (e.g., expanded).

IP/LP Geometry

A "good" formulation keeps this region small.

One measure of this is the Integrality Gap:

Integrality gap = max \(_{\text{instances}}\) \(\frac{(\text{IP}(I))/\text{LP}(I)}{2}\)

Unconstrained Facility Location Revisited

Given potential facility locations, customers to be served

- \(c_j\) = cost to build facility \(j\)
- \(h_{ij}\) = cost to meet all of customer \(i\)'s demand from facility \(j\)

\[
\text{min} \sum_j c_j x_j + \sum_{i,j} h_{ij} y_{ij}
\]

st. \(\sum_j y_{ij} = 1\) \(\forall i\) (each customer satisfied)

\(y_{i,j} = \begin{cases} 1 & \text{if customer } i \text{ is served by facility } j \\ 0 & \text{otherwise} \end{cases}\)

\(x_j, y_{i,j} \in [0,1]\)

Given potential facility locations, customers to be served

- \(c_j\) = cost to build facility \(j\)
- \(h_{ij}\) = cost to meet all of customer \(i\)'s demand from facility \(j\)

\[
\text{min} \sum_j c_j x_j + \sum_{i,j} h_{ij} y_{ij}
\]

st. \(\sum_j y_{ij} = 1\) \(\forall i\) (each customer satisfied)

\(y_{i,j} = \begin{cases} 1 & \text{if customer } i \text{ is served by facility } j \\ 0 & \text{otherwise} \end{cases}\)

\(x_j, y_{i,j} \in [0,1]\)
How the weaker LP “cheats”

Using \[ \sum_{j} y_{i} x_{j} \]

Allows the LP to completely satisfy customer i with facility j even with \( x_{j} = \frac{1}{n} \).

With these constraints:

\[ y_{i} x_{j} \leq x_{i} \forall i, j \]

If \( x_{j} = \frac{1}{n} \), then \( y_{i} \leq \frac{1}{n} \)

Can’t we just round the LP Solution?

• Not generally feasible
• If (miraculously) it is feasible, it’s not generally good

Example: Maximum Independent Set

• Find a maximum-size set of vertices that have no edges between any pair

\[ v_{i} = \begin{cases} 1 & \text{if vertex } i \text{ is in the MIS} \\ 0 & \text{otherwise} \end{cases} \]

\[ \max \sum v_{i} \]

s.t. \[ v_{i} + v_{j} \leq 1 \quad \forall (i, j) \in E \]

\[ v_{i} \in \{0, 1\} \]
Example: Maximum Independent Set

\[
\text{max } \sum_{i} v_i \\
\text{s.t. } v_i + v_j \leq 1 \quad \forall (i, j) \in E
\]

The zero-information solution \((v_i = 0.5 \text{ for all } i)\) is feasible and it’s optimal if the optimal MIS has size at most \(|V|/2\).

Rounding everything (up) is infeasible.

---

Can’t we project the lattice onto the objective gradient?

\[
c^T s = \text{opt}
\]

- Hard to find a feasible solution to project (NP-complete!)
- Make the objective a constraint and do binary search
- This is a lot harder in \(n\) dimensions than it looks like in 2

---

Perfect formulations

- Sometimes solving an LP is guaranteed to give an integer solution
  - All polytope corners have integer coefficients (naturally integer)
  - Sometimes only for specific objectives (e.g. \(c \cdot s \leq 0\))

---

Perfect Formulation Example: Minimum Cut

\[
\text{Capacity } u_e
\]

- Special nodes \(s\) and \(t\)
- Each edge \(e\) has capacity \(u_e\). Set of edges \(S\) has capacity \(\sum u_e\)
- Partition vertex set \(V\) into \(S, T\) where \(s \in S\) and \(t \in T\)
- A cut is the edges \((u, v)\) such that \(u \in S\) and \(v \in T\)

Find a cut with minimum capacity
Perfect Formulation Example: Minimum Cut IP

\[
\begin{align*}
\min & \quad \sum_{e} \gamma_e \\
\text{st} & \quad \gamma_e = 1 \text{ if } e \text{ is in the cut and 0 otherwise} \\
& \quad \gamma_e, v_e \geq 0 \\
& \quad v_e \in \{0, 1\}
\end{align*}
\]

The \( \gamma \) variables will be integral if the \( v \) variables are.

Total Unimodularity

The minimum cut matrix (possibly with slack variables) is totally unimodular (TU): all subdeterminants (including the matrix entries) have value 0, 1, or -1.

- All corner solutions \( x \) satisfy \( Ax = b \)
- By Kramer’s rule \( x \) will be integral

Network matrices (adjacency matrices of graphs) are TU.

Nemhauser and Wolsey (Integer and Combinatorial Optimization, Wiley, 1988) give some sufficient conditions for a matrix to be TU.

Note: if a matrix is TU, there is always an efficient combinatorial algorithm to solve the problem (not necessarily obvious)

Total Unimodularity is Fragile

- Example: Network Interdiction
  - Expend a limited budget to maximally damage the transport capacity of a network

Network Flow

- Source(s), sink (consumers) \( s, t \)
- Capacity (bottom number)
- Flow (top number)
- Maximum flow from \( s \) to \( t \) obeying
  - Capacity constraints on edges
  - Conservation constraints on all nodes other than \( s,t \)
Network Interdiction

- Each edge \( e \) now has a destruction cost \( d_e \) (cost to remove \( e \); assume linear)
- Budget \( B \)
  - Expend at most \( B \) removing (pieces of) edges in the network so resulting max flow is minimized

By LP duality (we’ll see later)
value of max flow = value of min cut
So
\[
\min_{\text{attacks}} \text{max flow} = \min_{\text{attacks}} \text{min cut}
\]
Pay to knock out transport capacity from \( s \) to \( t \)

A Mixed Integer Program for Network Inhibition

- Based on min-cut LP
- Find best cut to attack
- Decision variables place vertices on the \( s \) or \( t \) side as before
- All edges going across the cut must be destroyed (consume budget) or contribute to residual cut capacity

Network Inhibition IP

\[
\begin{align*}
\min \sum_{e} z_e \\
\text{s.t.} \quad y_e + z_e &\leq v_e \quad \forall (i,j) \\
& y_e + z_e = 1 \quad \forall (i,j) \\
& y_e + z_e \leq 1 \\
& \sum_{e} d_e z_e \leq B \\
& v_e \in \{0,1\}
\end{align*}
\]
Total Unimodularity is Fragile

\[ \min \sum u_i y_i \]

\[ \begin{align*}
& \text{s.t. } y_i + z_i = v_i - x_i \quad \forall (i,j) \\
& y_i + z_i = y_j - x_j \quad \forall (i,j) \\
& v_i = 0, v_i = 1 \\
& \sum d_{ij} = B \\
& y_i \in [0,1]
\end{align*} \]

The matrix is still TU without the budget constraint.
Adding the budget constraint makes the problem strongly NP-complete.
- No known polynomial-time approximation algorithms.
- Still has some very nice structure that gives a pseudo-approximation:
  - Pseudo-approximation might give a superoptimal solution that slightly exceeds the budget or it could give a true approximation.

Modeling Sets

Given a set \( T \):
- \( \sum x_i = 1 \) means select at least 1 element of \( T \)
  - Making sure at least one local warehouse has inventory for each customer.
- \( \sum x_i = 1 \) means select at most 1 element of \( T \)
  - Conflicts (e.g. modeled by a maximum independent set problem).
- \( \sum x_i = 1 \) means select exactly 1 element of \( T \)
  - Resource constraints.
- \( \sum x_i = 1 \) means select exactly 1 element of \( T \)
  - Time indexed scheduling variables \( s_{jt} \), schedule job \( j \) at time \( t \). This picks a single time for job \( j \).

Modeling Disjunctive Constraints

Let \( a_i x \leq b_i \) and \( a_i x \geq b_i \) be two constraints with nonnegative coefficients \( (a_i, b_i, i = 1, 2) \).

To force satisfaction of at least one of these constraints:

\[ \begin{align*}
& a_i x \leq b_i \\
& a_i x \geq (1-y)b_i \\
& y \in [0,1]
\end{align*} \]

Modeling Disjunctive Constraints - General Number

Let \( a_i x \leq b_i, i = 1, \ldots, m \) be \( m \) constraints with nonnegative coefficients \( (a_i, b_i) \).

To force satisfaction of at least \( k \) of these constraints:

\[ \begin{align*}
& a_i x \leq b_i, i = 1, \ldots, m \\
& \sum_{i=1}^{m} y_i \geq k \\
& y \in [0,1]
\end{align*} \]
Modeling a Restricted Set of Values

- Variable $x$ can take on only values in \( \{v_1, v_2, \ldots, v_m\} \)
  - Frequently the $v_i$ are sorted
  - Example: capacity of an airplane assigned to a flight

\[
x = \sum_{i=1}^{m} y_i v_i
\]
\[
\sum_{i=1}^{m} y_i = 1
\]
\[
y_i \in \{0,1\}
\]

- The $y_i$’s are a special ordered set.

Some simple logical constraints

Want \( y = x_1 \lor x_2 \) (logical or)
\[
y \in \{y_1, y_2\}
\]
Suffices if there is pressure in the objective function to keep $y$ low.

• Saw this in minimum cut

Similarly if we want \( y = x_1 \land x_2 \) (logical and)
\[
y \in \{y_1, y_2\}
\]
Suffices if there is pressure in the objective function to keep $y$ high.

Example: Protein Structure Comparison

- 2 nonadjacent amino acids share an edge if they’re physically close when folded

Contact Map

Example: Protein Structure Comparison

- 2 nonadjacent amino acids share an edge if they’re physically close folded
- Noncrossing alignment of two proteins to maximize shared contacts
- Measure of similarity
Protein Structure Comparison

- Variables $x_{ij} = 1$ if amino acid in position $i$ of the top protein is matched to amino acid in position $j$ of the bottom protein, 0 otherwise
- Helper variables $y_{ijkl} = x_{ij} \cdot x_{kl}$

Non-crossing alignment

- For any pair of edges, we can tell if they cross
- Helper variable $y_{ijkl} = 1$ if the pair is forbidden (simply don’t create this variable).
- There are more clever ways to do this (e.g., using Ramsey theory). See what you can come up with.

MIP Applications (Small Sample)

- Logistics
  - Capacity planning, scheduling, workforce planning, military spares management
- Infrastructure/network security
  - Vulnerability analysis, reinforcement, reliability, design, integrity of physical transport media
  - Sensor placement (water systems, roadways)
- Waste remediation
- Vehicle routing, fleet planning
- Bioinformatics: protein structure prediction/comparison, drug docking
- VLSI, robot design
- Tools for high-performance computing (scheduling, node allocation, domain decomposition, meshing)
Solving Integer Programs

- NP-hard
- Many special cases have efficient solutions or provably-good approximation bounds
  - Need time to explore structure
- General IPs can be hard due to size and/or structure

(Sufficiently) optimal solution is important
- When lives or big $ at stake
- For rigorous benchmarking of heuristic/approximation methods
- To gain structural insight for better algorithms/proofs.