Lecture 12

Integer linear optimization

Kin Cheong Sou
Department of Mathematical Sciences
Chalmers University of Technology and Göteborg University
December 9, 2014

We consider problems of the type

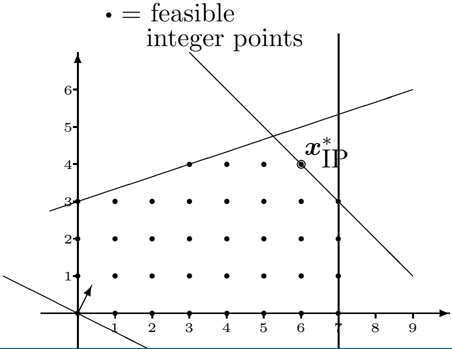
$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \in \mathbb{Z}^n \end{aligned} \tag{1}$$

That is, linear programs with additional **integrality requirements**.
Often we look at the special case of binary programs

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \in \{0,1\}^n \end{aligned} \tag{2}$$

Integer program

$$
\begin{array}{llrcrcll}
\max & z_{\text{IP}} = & x_1 & + & 2x_2 & & & \\
\text{s.t.} & & x_1 & + & x_2 & \leq & 10 & (1) \\
 & & -x_1 & + & 3x_2 & \leq & 9 & (2) \\
 & & x_1 & & & \leq & 7 & (3) \\
 & & & & x_1, x_2 & \geq & 0 & (4,5) \\
 & & & & x_1, x_2 & & \text{integer} &
\end{array}
$$



$$x^*_{\text{IP}} = \begin{pmatrix} 6 \\ 4 \end{pmatrix}$$

$$z^*_{\text{IP}} = 14$$

$$x^*_{\text{LP}} = \begin{pmatrix} 21/4 \\ 19/4 \end{pmatrix}$$

$$z^*_{\text{LP}} = 14\tfrac{3}{4} > z^*_{\text{IP}}$$

When are integer models needed/helpful?

- Products or raw materials are indivisible
- Logical constraints: "if A then B"; "A or B"
- Fixed costs
- Combinatorics (sequencing, allocation)
- On/off-decision to buy, invest, hire, generate electricity, ...

0-1 binary decision variables can model **logical decisions and relations**:

- 0-1 binary variables: $x = 1$ means "true"; $x = 0$ means "false".
- If x then y: $x \leq y$ ($x = 1 \implies y = 1$).
- "XOR": $x + y = 1$ (cannot be both "true" or both "false").
- Exactly one out of n must be true: $x_1 + x_2 + \ldots + x_n = 1$.
- At least 3 out of 5 must be chosen: $x_1 + x_2 + \ldots + x_5 \geq 3$.
- and more...

Disjoint feasible sets

Integer decision variables can model **disjoint feasible sets**:

- For example, either $0 \leq x \leq 1$ **or** $5 \leq x \leq 8$:

$$\begin{aligned} x &\geq 0 \\ x &\leq 8 \\ x &\leq 1 + 7y \\ x &\geq 5y \\ y &\in \{0, 1\} \end{aligned}$$

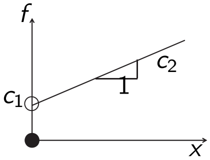
- Variable x may only take the values 2, 45, 78 or 107

$$\begin{aligned} &x = 2y_1 + 45y_2 + 78y_3 + 107y_4 \\ &y_1 + y_2 + y_3 + y_4 = 1 \\ &y_1, y_2, y_3, y_4 \in \{0, 1\} \end{aligned}$$

Fixed costs

- Want to minimize an objective function with **fixed cost**:

$$f(x) = \begin{cases} 0 & \text{if } x = 0, \\ c_1 + c_2 x & \text{if } 0 < x \leq M, \end{cases}$$



where $c_1 > 0$ is a fixed cost incurred as long as $x > 0$.

- Modeling fixed cost using binary decision variable:

$$\begin{aligned} f(x, y) &= c_1 y + c_2 x \\ x &\geq 0 \\ x &\leq My \\ y &\in \{0, 1\} \end{aligned}$$

- Fill a square $n \times n$ grid with numbers $1 \ldots n$
- Every number must occur exactly once in every row, column and box
- Huge number of reasonable configurations of numbers
- To the right is a supposedly very difficult sudoku

8	-	-	-	-	-	-	-	-
-	-	3	6	-	-	-	-	-
-	7	-	-	9	-	2	-	-
-	5	-	-	-	7	-	-	-
-	-	-	-	4	5	7	-	-
-	-	-	1	-	-	-	3	-
-	-	1	-	-	-	-	6	8
-	-	8	5	-	-	-	1	-
-	9	-	-	-	-	4	-	-

Sudoku cont.

Want to let $x_{ijk} = 1$ iff the solution to the puzzle puts number k at row i, column j. Let a_{ij} be the given values of the puzzle we want to solve for $(i, j) \in \mathcal{D}$.

$$\begin{aligned}
\text{minimize} \quad & \mathbf{c}^{\mathrm{T}}\mathbf{x} \\
\text{subject to} \quad & \sum_{j=1}^{n} x_{ijk} = 1, && i, k = 1, \dots, n, && (1) \\
& \sum_{i=1}^{n} x_{ijk} = 1, && j, k = 1, \dots, n && (2) \\
\sum_{i=m(s-1)+1}^{ms} & \sum_{j=m(p-1)+1}^{mp} x_{ijk} = 1, && s, p = 1, \dots, m, k = 1, \dots, n, && (3) \\
& \sum_{k=1}^{n} x_{ijk} = 1, && i, j = 1, \dots, n, && (4) \\
& x_{ijk} = 1, && (i, j) \in \mathcal{D}, k = a_{ij}, && (5) \\
& x_{ijk} \in \{0, 1\}, && i, j, k = 1, \dots, n. && (6)
\end{aligned}$$

Sudoku cont

- (1)–(3) force every number to be used once in each row, column, and box.
- (4) forces each position to use exactly one number.
- (5) forces our solution to agree with the initial data.
- (6) Variables must be binary.
- The objective function lets me tune which solution I want to get.

Solution:

```
0.02 s
208 MIP simplex iterations
5 branch-and-bound nodes
```

8	1	2	7	5	3	6	4	9
9	4	3	6	8	2	1	7	5
6	7	5	4	9	1	2	8	3
1	5	4	2	3	7	8	9	6
3	6	9	8	4	5	7	2	1
2	8	7	1	6	9	5	3	4
5	2	1	9	7	4	3	6	8
4	3	8	5	2	6	9	1	7
7	9	6	3	1	8	4	5	2

Other applications of integer optimization

- Facility location (new hospitals, shopping centers, etc.)
- Scheduling (on machines, personnel, projects, schools)
- Logistics (material- and warehouse control, vehicle routing)
- Distribution (transportation of goods, buses for disabled persons)
- Production planning
- Telecommunication (network design, frequency allocation)

- In a sense no. For binary programs (2) we could in principle enumerate all 2^n possible solutions.
- The more general case (1) is not as straightforward, but clever finite enumerative schemes exist.
- However, integer programming is **NP-hard**, meaning that is unlikely that a polynomial time algorithm exists. Computation cost grows very rapidly with problem size.

Assign n persons to carry out n jobs $\qquad$ $\#$ feasible solutions: $n!$

Assume that a feasible solution is evaluated in 10^{-9} seconds

n	2	5	8	10	100
$n!$	2	120	$4.0 \cdot 10^4$	$3.6 \cdot 10^6$	$9.3 \cdot 10^{157}$
$\lceil \text{time} \rceil$	10^{-8} s	10^{-6} s	10^{-4} s	10^{-2} s	10^{142} yrs

Complete enumeration of all solutions is **not** an efficient algorithm!

An algorithm exists that solves this problem in time $\mathcal{O}(n^3) \propto n^3$

n	2	5	8	10	100	1000
n^3	8	125	512	10^3	10^6	10^9
$\lceil \text{time} \rceil$	10^{-8} s	10^{-7} s	10^{-6} s	$10^{-6} s$	10^{-3} s	1 s

Solution methods, overview

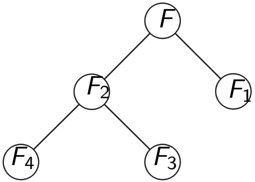
- General solution method (can be expensive but general)
  - Branch and bound method (divide-and-conquer)
  - Cutting plane method (polyhedral approximation)
  - Dynamic programming (divide-and-conquer)
  - Algebraic method (e.g., Graver bases)
- Exact solution method for special cases (efficient but not general)
  - Shortest path problem
  - Minimum cut problem
  - Minimum spanning tree problem
  - Bipartite matching problem
  - Assignment problem and more...
- Approximate solution methods
  - Usually more efficient; may or may not have error bounds

Branch and bound method, I

- Divide feasible set F into $F_1, F_2, \dots, F_k$.

Instead of solving $\begin{array}{ll} \min\limits_x & c^T x \\ \text{s.t.} & x \in F, \end{array}$ solve for all i $\begin{array}{ll} \min\limits_x & c^T x \\ \text{s.t.} & x \in F_i. \end{array}$

- May need to recursively divide F_i, $i = 1, \dots, k$. This is **branching**.



- Dividing F all the way to singletons $\rightarrow$ enumeration. Is it necessary?

Branch and bound method, II

Do we always need to divide F_i further when considering

$$\text{subproblem with } F_i\text{:} \quad \begin{array}{ll} \min\limits_{x} & c^T x \\ \text{s.t.} & x \in F_i \end{array} \quad ?$$

We can stop further dividing F_i, if one of the following holds:

- F_i is an empty set.
- Somehow we manage to solve subproblem with F_i to optimality. In this case we possibly update "the current best" objective value z_{best}.
- **Bounding:** If we find $b(F_i)$, a lower bound of optimal objective value of subproblem with F_i, such that

$$b(F_i) \geq z_{\text{best}}.$$

BNB performance depends critically on quality of lower bound!

Bounding, LP relaxation

How to check if $F_i = \emptyset$? How to find lower bound $b(F_i)$?

- Let F_i be feasible set of the IP (integer program) below, with LP relaxation:

$$
\text{(IP)} \quad \begin{array}{rll} z^*_{\text{IP}} = & \min\limits_x & c^T x \\ & \text{s.t.} & Ax \geq b \\ & & Dx \geq d \\ & & x \text{ integer} \end{array}
\qquad\qquad
\text{(LP)} \quad \begin{array}{rll} z^*_{\text{LP}} = & \min\limits_x & c^T x \\ & \text{s.t.} & Ax \geq b \\ & & Dx \geq d \\ & & x \text{ real} \end{array}
$$

- (LP) is a relaxation of (IP), since feasible set of (LP) includes (IP)'s.
- If (LP) is infeasible then (IP) is infeasible, meaning that $F_i = \emptyset$.
- $z^*_{\text{LP}} \leq z^*_{\text{IP}}$. Thus, can set lower bound as $b(F_i) = z^*_{\text{LP}}$.
- If solving (LP) yields integer solution, then it is optimal to (IP) too.

- For IP problem with feasible set F_i:

$$z^*_{\mathrm{IP}} = \begin{array}{rl} \min\limits_{x} & c^T x \\ \text{s.t.} & Ax \geq b \\ & Dx \geq d \\ & x \text{ integer} \end{array}$$

- Can also obtain lower bound $b(F_i)$ by "dualizing" some constraints:

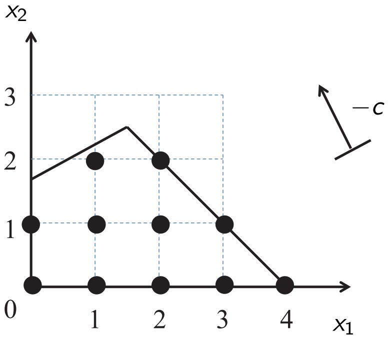
$$z^*_{\mathrm{LD}} = \begin{array}{rl} \max\limits_{\mu} & q(\mu) \\ \text{s.t.} & \mu \geq \mathbf{0} \end{array} \quad \text{with} \quad q(\mu) = \begin{array}{rl} \min\limits_{x} & c^T x + \mu^T (b - Ax) \\ \text{s.t.} & Dx \geq d, \; x \text{ integer} \end{array}$$

- Method is practical only when $q(\mu)$ is easy to evaluate.
- $z^*_{\mathrm{LP}} \leq z^*_{\mathrm{LD}} \leq z^*_{\mathrm{IP}}$ – lower bound by Lagrangian dual is always no worse than LP relaxation bound. Inequalities can be strict.

Branch and bound, illustration (1)

- An example linear integer programming problem:

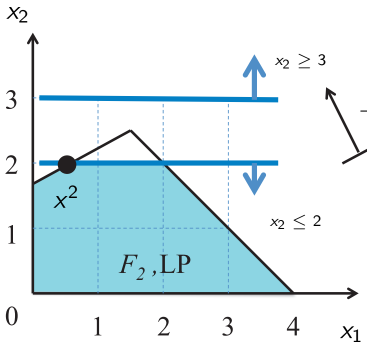
$$\begin{aligned} \text{minimize} \quad & x_1 - 2x_2 \\ \text{subject to} \quad & -4x_1 + 6x_2 \leq 9 \\ & x_1 + x_2 \leq 4 \\ & x_1, x_2 \geq 0 \\ & x_1, x_2 \text{ integer} \end{aligned}$$

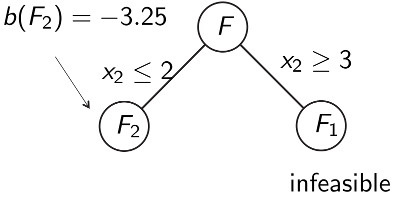


- Dots are (integer) feasible points. Let S denote feasible set.

Branch and bound, illustration (2)

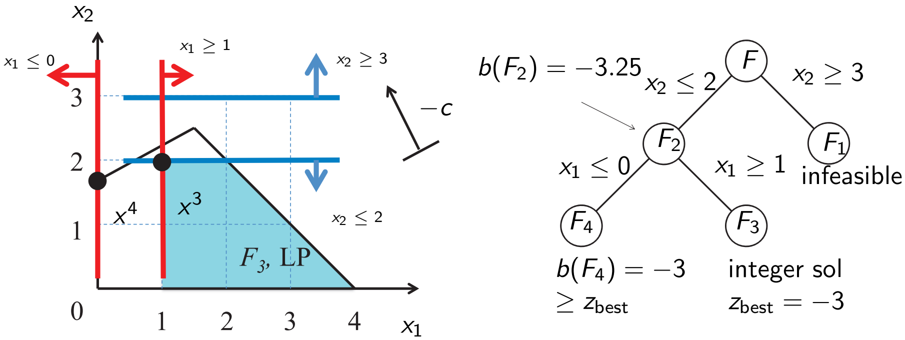
- F is divided into $F_1 = \{x \mid x_2 \geq 3\} \cap S$ and $F_2 = \{x \mid x_2 \leq 2\} \cap S$.
- $F_1 = \emptyset$. No need to consider further.
- F_2: LP relaxation $x^2 = (0.75, 2)$, lower bound $b(F_2) = -3.25$.
- Split F_2: $F_3 = \{x | x_1 \geq 1, x_2 \leq 2\} \cap S$, $F_4 = \{x | x_1 \leq 1, x_2 \leq 2\} \cap S$.





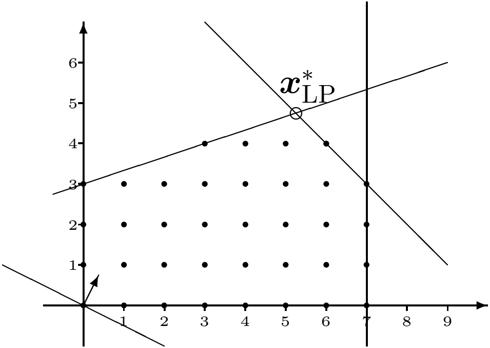
Branch and bound, illustration (3)

- Split F_2: $F_3 = \{x \mid x_1 \geq 1, x_2 \leq 2\} \cap S$, $F_4 = \{x \mid x_1 \leq 1, x_2 \leq 2\} \cap S$
- F_3: LP relaxation $x^3 = (1, 2)$, integer valued! Update $z_{\text{best}} = -3$.
- F_4: LP relaxation $x^4 = (0, 1.5)$, $b(F_4) = -3 \geq z_{\text{best}}$, so remove F_4.



Cutting plane

- LP relaxation has too large feasible set...
- Add cuts (i.e., valid inequalities satisfied by all IP feasible solutions but not LP relaxation solutions) to tighten the relaxation.
- We need one in this example. Which one?......... answer is $x_2 \leq 4$.



A fundamental theorem for MILP

What is the tightest LP relaxation? How good is it? The answer is...

$$\text{(IP)} \quad \begin{array}{ll} \min\limits_{x} & c^T x \\ \text{s.t.} & s \in S, \end{array} \qquad\qquad \text{(R)} \quad \begin{array}{ll} \min\limits_{x} & c^T x \\ \text{s.t.} & s \in \text{conv}(S). \end{array}$$

- (R) is a relaxation of (IP) as $S \subset \text{conv}(S)$, but is (R) LP relaxation?

> Let $\mathbf{A}$ be a rational matrix, $\mathbf{b}$ a rational vector, and let $S = \{\mathbf{x} \in \mathbb{Z}^n \,|\, \mathbf{Ax} \leq \mathbf{b}\}$. Then the convex hull of S, denoted $\text{conv}(S)$, is a polyhedron. Also, the extreme points of $\text{conv}(S)$ belong to S.

- By the thm, (R) is LP relaxation of (IP), as $\text{conv}(S)$ is a polyhedron.
- (R) is tightest: solving (R) using simplex method also solves (IP).
- But, representation of $\text{conv}(S)$ is difficult to find...

Fundamental theorem for MILP, proof

Let $\mathbf{A}$ be a **rational** matrix, $\mathbf{b}$ a **rational** vector, and let $S = \{\mathbf{x} \in \mathbb{Z}^n \mid \mathbf{Ax} \leq \mathbf{b}\}$. Then the convex hull of S, denoted $\text{conv}(S)$, is a polyhedron. Also, the extreme points of $\text{conv}(S)$ belong to S.

Proof:

- $\text{conv}(S)$ need not be a polyhedron if A is not rational. For example, $S = P \cap \mathbb{Z}^n$ and $P = \{x_1 \geq 0, x_2 \geq 0, x_2 \leq \sqrt{2}x_1\}$.
  - P is only candidate for $\text{conv}(S)$, but $(n, \sqrt{2}n) \notin S$ for all $n \in \mathbb{N}$ except when $n = 0$.

Cutting plane methods

- Integer program solution method by building better and better polyhedral approximations of the convex hull of IP feasible set S. For polyhedral (outer) approximation F_i such that $S \subset F_i$, solve

$$\text{LP relaxation with } F_i: \quad \begin{array}{ll} \underset{x}{\text{minimize}} & c^T x \\ \text{subject to} & s \in F_i. \end{array}$$

- Let x^{LP} solve LP relaxation. If $x^{\text{LP}} \in S$, then we are done.
- Otherwise, generate a cut of the form $a^T x \leq d$ such that

$$a^T x^{\text{LP}} > d \quad \text{but} \quad a^T x \leq d \quad \forall x \in S.$$

- Update polyhedral approximation $F_{i+1} \leftarrow F_i \cap \{x \mid a^T x \leq d\}$. Solve updated LP relaxation with F_{i+1}.
- Cutting plane method performance depends critically on “depth” of the cut!

Generating a cut

- Assume polyhedral approximation $F_i = \{x \mid Ax = b, x \geq 0\}$
- Let $x^{\text{LP}} \in \operatorname*{argmin}_{x \in F_i} \; c^T x$, and assume x^{LP} is not integral
- Re-arrange A: B opt basis of x^{LP}; N index set of nonbasic variables
- Denote $\bar{a}_{pq} = (B^{-1}A_q)_p$, $\bar{a}_{p0} = (B^{-1}b)_p$
- At least one of $\bar{a}_{p0}$ is not integral; let $\bar{a}_{j0} \notin \mathbb{Z}$
- $x \in F_i \overbrace{\Longrightarrow}^{(B^{-1}Ax)_j = (B^{-1}b)_j} x_j + \sum_{k \in N} \bar{a}_{jk} x_k = \bar{a}_{j0} \overbrace{\Longrightarrow}^{x \geq 0} x_j + \sum_{k \in N} \lfloor \bar{a}_{jk} \rfloor x_k \leq \bar{a}_{j0}$
- $x \in F_i \cap \mathbb{Z}^n \implies x_j + \sum_{k \in N} \lfloor \bar{a}_{jk} \rfloor x_k \leq \lfloor \bar{a}_{j0} \rfloor$
- But, $x_j^{\text{LP}} + \sum_{k \in N} \lfloor \bar{a}_{jk} \rfloor x_k^{\text{LP}} = x_j^{\text{LP}} = \bar{a}_{j0} > \lfloor \bar{a}_{j0} \rfloor$
- Thus, $x_j + \sum_{k \in N} \lfloor \bar{a}_{jk} \rfloor x_k \leq \lfloor \bar{a}_{j0} \rfloor$ is a desired cut

Approximate solution methods

- Branch and bound and cutting plane methods provide exact optimal solution, but sometimes we don't want to wait too long
- We can resort to approximate solution methods:
  - LP relaxation might not provide integer optimal solutions, but we can "round" them to integer feasible solutions.
  - Lagrangian dual relaxation might not provide feasible solutions, but from there we can construct suboptimal feasible solutions.
  - Randomized algorithms (e.g., genetic algorithms, simulated annealing) compare objective values at randomly chosen feasible solutions – not much theoretical guarantee but empirically they might find good suboptimal solutions.