Numerical Optimization

Lecture 8: Integer Programming

Hao Wang
Email: wanghao1@shanghaitech.edu.cn

Contents

Contents

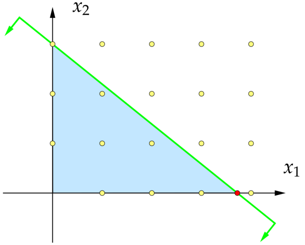
Problem formulation

Integer Programming (IP) problems:

$$
\begin{array}{rlllll}
\min & c_1 x_1 & +c_2 x_2 & +\cdots & +c_n x_n & \\
\text{s.t.} & a_{11} x_1 & +a_{12} x_2 & +\cdots & +a_{1n} x_n & \leq b_1 \\
& a_{21} x_1 & +a_{22} x_2 & +\cdots & +a_{2n} x_n & \leq b_2 \\
& \vdots & & & & \\
& a_{n1} x_1 & +a_{n2} x_2 & +\cdots & +a_{nn} x_n & \leq b_n \\
& & & & x_i & \in \mathbb{Z},\ \forall i \in \{1, 2, \ldots, n\}.
\end{array}
$$

Discontinuous

Why can't we just round numbers up/down?



$$\begin{array}{lll} \max & x_1 & +1.1x_2 \\ & \frac{x_1}{3.7} & +\frac{x_2}{3} \leq 1 \\ & & x_1, x_2 \in \mathbb{Z}_+ \end{array}$$

- Optimal solution of the LP relaxation: $(3.7, 0)$, obj. f.: 3.7
- Rounded solution: $(3, 0)$, obj. f.:3.0
- Optimal solution of the original problem: $(0, 3)$, obj. f.: 3.3

LP and IP

IP is a bit younger than LP, but is the subject of extensive research and can model countless problems in Industry:

- Airline crew scheduling
- Vehicle Routing
- Financial applications
- Design of Telecommunications networks

IP problems are difficult to solve:

- Require very specialized techniques (some use LP relaxations to find lower bounds)
- A good model makes a problem easier (but not easy)

i.e. unlike LP, the way we model an Optimization problem **affects** the chances to solve it

Contents

Binary variables, logical operators

- model yes/no decisions: $x_i \in \{0, 1\}$
- $x_i = 0$ if the decision is “no”
- $x_i = 1$ if it is “yes”
- can use logical operators: implications, disjunctions, etc.:
  - 小红 or 小强 will have ice cream, but not both:

    $$x_{\text{小红}} + x_{\text{小强}} \leq 1$$

  - At least one among 小红 and 小强 will have ice cream:

    $$x_{\text{小红}} + x_{\text{小强}} \geq 1$$

  - If 小红 has ice cream, then 小刚 will have one too:

    $$x_{\text{小红}} \leq x_{\text{小刚}}$$

  - 小强 gets ice cream if and only if 小明 does not get any:

    $$x_{\text{小强}} = 1 - x_{\text{小明}}$$

Binary variables and operations with sets

Binary variables are useful to model problems on sets. E.g.:

- Choose a subset S of a set A of elements such that S has certain properties (e.g. not more than K elements, etc.)
- Each element $i \in A$ has a cost c_a
  $\implies$ The cost of a solution S is $\sum_{i \in S} c_i$
- Define variable x_i:

$$x_i = \begin{cases} 1 & \text{if} \quad i \in S \\ 0 & \text{otherwise} \end{cases}$$

- now the cost of a solution S is $\sum_{i \in A: x_i = 1} c_i = \sum_{i \in A} c_i x_i$
- define properties similarly, e.g. $|S| \leq K$ is $\sum_{i \in A} x_i \leq K$

Example: Subset Sum

Two brothers, 小明 and 小强, inherit from their dear uncle a set A of antique objects.

- Each worth a lot of money, c_i for all $i \in A$
- They want to share these objects in a balanced manner
  $\implies$ minimize the difference between their total values
- $S_L \subset A$ contains the objects that 小明 will get, while
  $S_J \subset A \setminus S_L$ are the remaining objects
  $\implies$ minimize

$$\left|\sum_{i \in S_L} c_i - \sum_{j \in S_J} c_j\right|$$

How to model this with IP?

Example: Subset Sum (cont'd)

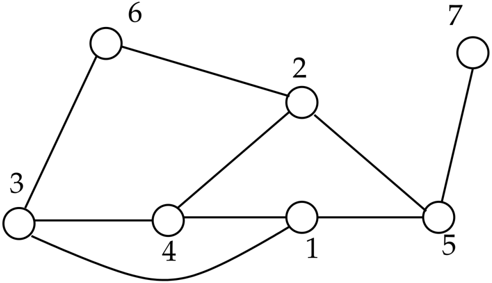
$$x_i = \begin{cases} 1 & \text{if 小明 gets the i-th object} \\ 0 & \text{if 小强 gets the i-th object} \end{cases}$$

Then the integer model is

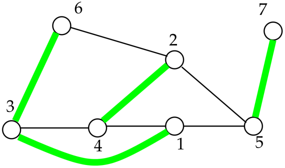
$$\begin{aligned} \min \quad & \left| \sum_{i \in A} c_i x_i - \sum_{i \in A} c_i (1 - x_i) \right| \\ \text{s.t.} \quad & x_i \in \{0, 1\}, \ \forall i \in A \end{aligned}$$

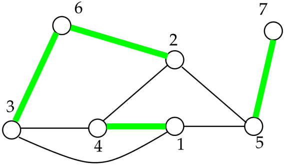
Example: the edge covering problem

In a graph $G = (V, E)$ as in the figure, choose a subset S of edges such that all nodes are "covered" by at least one edge in S. Minimize the number of edges used



Example: the edge covering problem





Edge covering

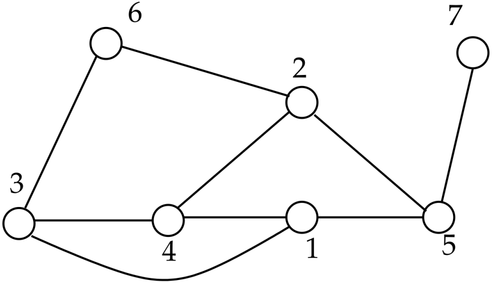
$$
\begin{array}{rlllllllllll}
\min & x_{13}+ & x_{14}+x_{15}+ & x_{24}+x_{25}+ & x_{26}+x_{34}+ & x_{36}+x_{57} & \\
\text{s.t.} & x_{13}+ & x_{14}+x_{15} & & & & \geq 1 \\
& & & x_{24}+x_{25}+ & x_{26} & & \geq 1 \\
& x_{13}+ & & & x_{34}+ & x_{36} & \geq 1 \\
& & x_{14}+ & x_{24}+ & x_{34}+ & & \geq 1 \\
& & x_{15}+ & x_{25}+ & & +x_{57} & \geq 1 \\
& & & & x_{26}+ & x_{36} & \geq 1 \\
& & & & & x_{57} & \geq 1 \\
& x_{13}, & x_{14},x_{15}, & x_{24},x_{25}, & x_{26},x_{34}, & x_{36},x_{57} & \in\{0,1\}
\end{array}
$$

Edge covering

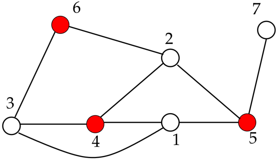
$$\begin{aligned} \min \quad & \sum_{\{i,j\} \in E} x_{\{i,j\}} & \\ \text{s.t.} \quad & \sum_{j \in V : \{i,j\} \in E} x_{\{i,j\}} \geq 1 & \forall i \in V \\ & x_{\{i,j\}} \in \{0,1\} & \forall \{i,j\} \in E \end{aligned}$$

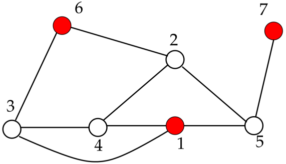
Example: the node packing (or stable set) problem

In a graph $G = (V, E)$ as in the figure, choose a subset S of nodes such that no two nodes i and j in S are adjacent, i.e. share an edge $\{i, j\}$. In other words, if both i and j are included in S, then there must be no edge $\{i, j\}$. Maximize the number of nodes used.



Example: the node packing (or stable set) problem





Node packing

$$
\begin{array}{rlllllll}
\max & x_1 + & x_2 + x_3 + & x_4 + x_5 + & x_6 + x_7 & \\
\text{s.t.} & x_1 + & x_3 & & & \leq 1 \\
& x_1 + & & x_4 & & \leq 1 \\
& x_1 + & & x_5 & & \leq 1 \\
& & x_2 + & x_4 & & \leq 1 \\
& & x_2 + & x_5 & & \leq 1 \\
& & x_2 + & & x_6 & \leq 1 \\
& & x_3 + & x_4 & & \leq 1 \\
& & x_3 + & & x_6 & \leq 1 \\
& & & x_5 + & x_7 & \leq 1 \\
& x_1, & x_2, x_3, & x_4, x_5, & x_6, x_7 & \in \{0, 1\}
\end{array}
$$

Node packing

$$
\begin{aligned}
\min \quad & \sum_{i \in V} x_i & \\
\text{s.t.} \quad & x_i + x_j \leq 1 & \forall \{i, j\} \in E \\
& x_i \in \{0, 1\} & \forall i \in V
\end{aligned}
$$

Switching constraints on/off

Suppose constraint $a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq b$, or $a^Tx \leq b$ for short, depends on the value a binary variable y.

- for instance, $y = 1 \Leftrightarrow a^Tx \leq b$ with $y \in \{0, 1\}$

How do we model this in IP? The constraint can still be linear, but it will depend on y.

Switching constraints on/off

the equivalence: $y = 1 \Leftrightarrow a^T x \leq b$. It means

$$y = 1 \implies a^T x \leq b \text{ and } a^T x \leq b \implies y = 1$$

or equivalently

$$y = 1 \implies a^T x \leq b \text{ and } y = 0 \implies a^T x > b$$

We can model the first as $a^T x \leq b + M(1 - y)$; the second is:

$$a^T x > b - My$$

In general, the "$>$" or "$<$" are not accepted, as it depends on the precision of solver/modeler/computer (e.g. 10^{-15}). Same for "$\neq$". Use small numbers:

$$a^T x \geq b + \epsilon - My$$

What are good values of M and ϵ?

$$a^T x \leq b + M(1 - y) \quad a^T x \geq b + \epsilon - My$$

- Suppose each variable x_i has lower/upper bounds $l_i \leq x_i \leq u_i$. For short, let's use $l \leq x \leq u$ or $x \in [l, u]$
- Otherwise, use huge bounds $-10^{20} \leq x_i \leq 10^{20}$ (bad!)
- $a^T x \leq +\infty$ means "$a^T x$ is at most the maximum value it can take with $x \in [l, u]$" (redundant: $a^T x$ can be anything)
- $a^T x \geq -\infty$ means "$a^T x$ is at least the minimum value it can take with $x \in [l, u]$" (likewise)
- If $a \geq 0$, i.e. if all $a_i \geq 0$, then $M = au$ and $M = al$

$\implies$ M is $\sum_{i:a_i \geq 0} a_i u_i + \sum_{i:a_i \leq 0} a_i l_i - b$
$\implies$ M is $b - (\sum_{i:a_i < 0} a_i u_i + \sum_{i:a_i > 0} a_i l_i)$

Example

For the case

$$\begin{aligned} y = 1 \Leftrightarrow 3x_1 + 5x_2 - 2x_3 &\leq 6 \\ -7 \leq x_1 &\leq 4 \\ 0 \leq x_2 &\leq 12 \\ -11 \leq x_3 &\leq -1 \\ y &\in \{0, 1\} \end{aligned}$$

becomes

$$3x_1 + 5x_2 - 2x_3 \leq 6 + M(1 - y)$$
$$3x_1 + 5x_2 - 2x_3 \geq 6 + \epsilon - My$$

$$M = 3 \cdot 4 + 5 \cdot 12 - 2 \cdot (-11) - 6 = 88$$
$$M = 6 + \epsilon - [3 \cdot (-7) + 5 \cdot 0 - 2 \cdot (-1)] = 25 + \epsilon$$

which means

$$3x_1 + 5x_2 - 2x_3 \begin{cases} \leq 6 & \text{if } y = 1 \\ \geq 6 + \epsilon & \text{if } y = 0 \end{cases}$$

Contents

Relaxations and efficiency

Integer programming problems:

$$
\begin{array}{lllllll}
(IP) & \min & c_1x_1 & +c_2x_2 & \ldots & +c_nx_n & \\
 & & a_{11}x_1 & +a_{12}x_2 & \ldots & +a_{1n}x_n & \leq b_1 \\
 & & a_{21}x_1 & +a_{22}x_2 & \ldots & +a_{2n}x_n & \leq b_2 \\
 & & \vdots & & & & \\
 & & a_{m1}x_1 & +a_{m2}x_2 & \ldots & +a_{mn}x_n & \leq b_m \\
 & & & & & x_i \in \mathbb{Z} & \forall i \in J \subseteq \{1, 2 \ldots, n\}
\end{array}
$$

Relaxations and efficiency

Or, for short,

$$\begin{aligned} (IP) \quad \min \quad & c^T x \\ \text{s.t.} \quad & Ax \le b \\ & x_i \in \mathbb{Z} \quad \forall i \in \{1, 2, \cdots, n\} \end{aligned}$$

can be solved using their LP relaxation:

$$\begin{aligned} (LP) \quad \min \quad & c^T x \\ \text{s.t.} \quad & Ax \le b \end{aligned}$$

A global optimum z of (LP) is a lower bound for (IP).

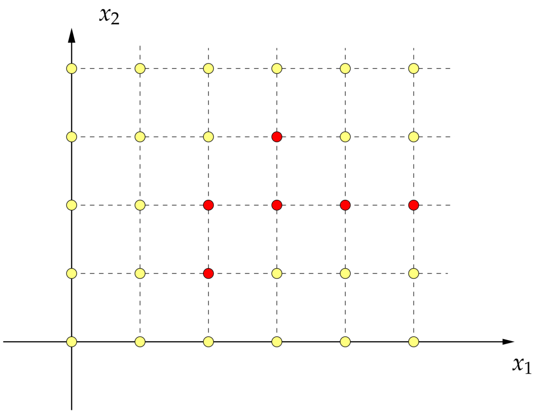
Relaxations and efficiency

- If an optimal solution x^* of (LP) is feasible for (IP), i.e., for all $i \in J$ we have $x_i^* \in \mathbb{Z}$, we're done!
- This is not the case, usually. . .
- What do we know about the optimal solutions of (LP)? They are all vertices of the polyhedron

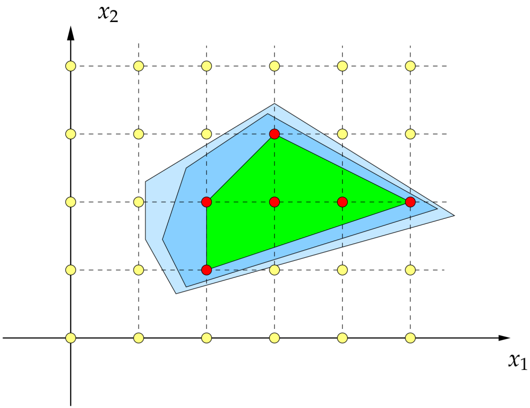
$$\{x \in \mathbb{R}^n : Ax \leq b\}$$

- Therefore, it would be just great if all vertices of (LP) were feasible for (IP). Solving IPs would amount to solving LPs, which are a lot easier.
- A good model may not achieve just that, but it can help a lot.

Relaxations, the geometrical standpoint



Relaxations, the geometrical standpoint



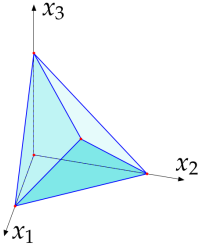
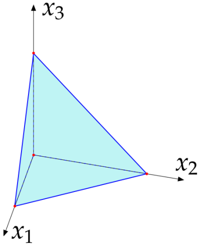
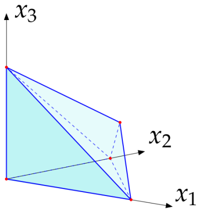
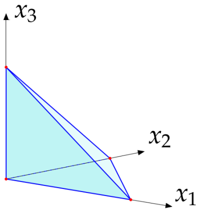
Relaxations: the clique inequality

Two models for one problem have the same feasible set and global optima, but may be solved differently:

$$\left.\begin{array}{rl} P_1 : \min & -7x_1 - 8x_2 - 9x_3 \\ \text{s.t.} & x_1 + x_2 \leq 1 \\ & x_1 + x_3 \leq 1 \\ & x_2 + x_3 \leq 1 \\ & x_1, x_2, x_3 \in \{0,1\} \end{array}\right\} \equiv \left\{ \begin{array}{rl} P_2 : \min & -7x_1 - 8x_2 - 9x_3 \\ \text{s.t.} & x_1 + x_2 + x_3 \leq 1 \\ & x_1, x_2, x_3 \in \{0,1\} \end{array}\right.$$

Relaxations: the clique inequality

- Consider relaxations R_1, R_2 of P_1, P_2 with $x_i \in [0, 1]$.
- R_1: optimal soln. $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$, obj.f. -12: lower bound for P_1, P_2
- R_2: optimal solution $(0, 0, 1)$, obj.f. -9: lower bound for P_2 and P_1, and feasible for P_2 and P_1!
- $\implies$ optimum of P_1, P_2: -9, and P_2 is a better model than P_1

R_1	R_2
	
	

Good vs. bad models: Uncapacitated Facility Location

A set J of retailers has to be served by a set S of plants, yet to be built. We don't know where the plants will be, but there is a set I of potential sites, and there is

- a cost f_i for building plant $i \in I$
- a (transportation) cost c_{ij} from plant i to retailer j

Each retailer will be served by exactly one plant

Choose a subset S of I such that the total cost is minimized.

Variables:

- x_i, $i \in I$: 1 if plant i is built, 0 otherwise
- y_{ij} assigns retailer j to plant i: 1 if i serves retailer j, 0 otherwise

Contents

How do we solve an Integer Programming problem?

$$
\begin{aligned}
(IP) \quad \min \quad & c^T x \\
\text{s.t.} \quad & Ax \leq b \\
& x_i \in \mathbb{Z} \quad \forall i \in \{1, 2, \cdots, n\}
\end{aligned}
$$

- Relaxing integrality gives an LP problem (easy)
- Solving LP gives a lower bound
- But the solution x^* may have fractional component $x_i^* \notin \mathbb{Z}$ with $i \in J$, infeasible for (P).

How do we solve an Integer Programming problem?

Divide the solution set, partition the problem into two new subproblems, P_1 and P_2 , with

$$P_1 : x_i \leq \lfloor x_i^* \rfloor \quad P_2 : x_i \geq \lceil x_i^* \rceil$$

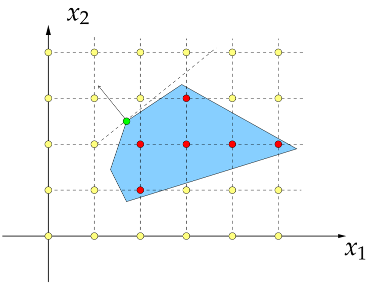
(P_1: $x_i \leq 3$ and P_2 : $x_i \geq 4$) and recursively solve P_1 and P_2 .

- Good: no feasible solution of P_1 or P_2 has $x_i = 3.31$

The Branch & Bound - Devide and concour

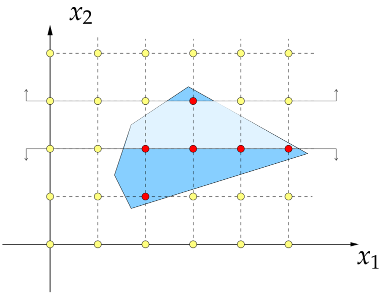
- If we solve the LP relaxation of P_1 and find a fractional point, we can recursively branch on P_1 and obtain two new subproblems P_3 and P_4.
- In principle, we have to branch on any node P_k unless its LP relaxation returns a feasible solution or it is infeasible.

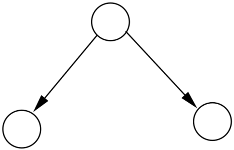
Example: minimize $11x_1 - 10x_2$



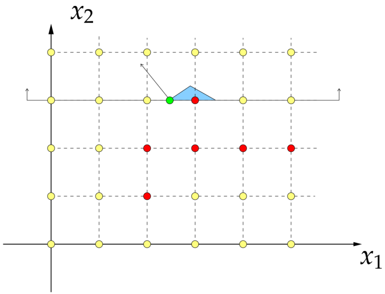
$lb = -9.2$
$ub = ?$

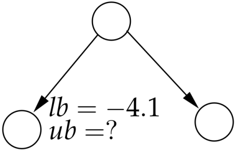
Example: minimize $11x_1 - 10x_2$



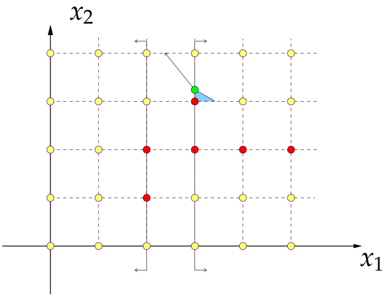


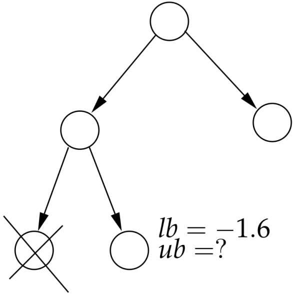
Example: minimize $11x_1 - 10x_2$



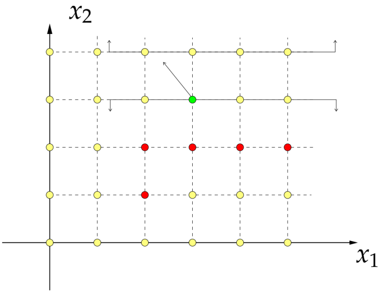


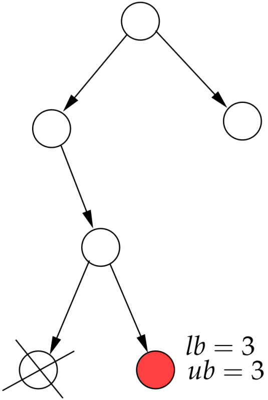
Example: minimize $11x_1 - 10x_2$



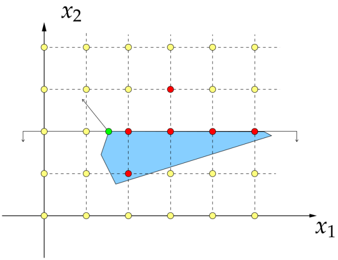


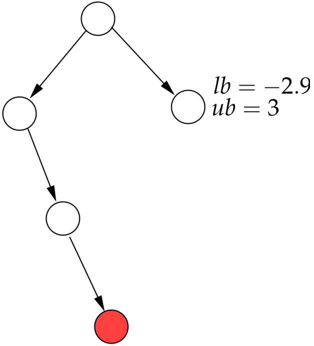
Example: minimize $11x_1 - 10x_2$



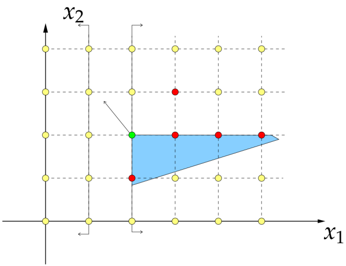


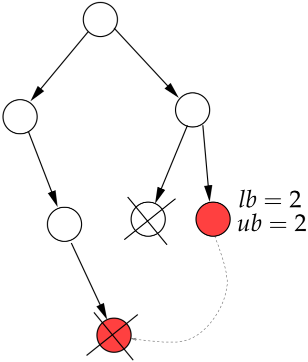
Example: minimize $11x_1 - 10x_2$





Example: minimize $11x_1 - 10x_2$





The Bound in Branch & Bound

In practice, the upper bound is very useful! Suppose you just found an **upper** bound of 194.

- P_3 has a **lower** bound of 146, P_4 of 203
- 203 is a **lower** bound for $P_4 \implies$ any feasible solution of P_4 has objective function value worse than 203 (i.e., ≥ 203)
- We already have something better $(194) \implies$ discard P_4 How to find upper bounds?