Lecture 8 Constrained optimization and integer programming

Weinan E[1,2] and Tiejun Li[2]

[1]Department of Mathematics,
Princeton University,
weinan@princeton.edu

[2]School of Mathematical Sciences,
Peking University,
tieli@pku.edu.cn
No.1 Science Building, 1575

Outline

Constrained optimization

- Suppose an investor own a block of S shares that we want to sell over the next N days. The total expected value of our shares is

$$V(\boldsymbol{s}) = \sum_{t=1}^{N} p_t s_t$$

where $(s_1, \cdots, s_N)$ is the amount that we sell on each day and $(p_1, \cdots, p_N)$ are the prices on each day. Moreover, the price p_t follows the following dynamics

$$p_t = p_{t-1} + \alpha s_t, \quad t = 1, \cdots, N$$

How should the investor sell his block of shares ?

- Mathematical formulation:

$$\max \sum_{t=1}^{N} p_t s_t$$

Subject to the constraint $\sum_{t=1}^{N} s_t = S, \quad p_t = p_{t-1} + \alpha s_t, \quad s_t \geq 0, \quad t = 1, \cdots, N;$

- A constrained nonlinear optimization.

0-1 Knapsack problem

- The thief wants to steal n items. The i-th item weights w_i and has value v_i. The problem is to take most valuable load with limit of weight W.
- Mathematical formulation:

$$\max V = \sum_{j=1}^{n} v_j x_j$$

$$\sum_{j=1}^{n} w_j x_j \leq W$$

$$x_j = 0 \text{ or } 1, \quad j = 1, \ldots, n$$

- x_j must be integers. An integer programming problem.

Assignment problem

- Assign n persons to finish n jobs. The cost for the i-th person to do j-th job is c_{ij}. Find the optimal assignment procedure to minimize the cose.
- Mathematical formulation: Define $x_{ij} = 1$ if the i-th person does j-th job, and $x_{ij} = 0$ otherwise, then

$$\max z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$\sum_{j=1}^{n} x_{ij} = 1, i = 1, \ldots, n$$

$$\sum_{i=1}^{n} x_{ij} = 1, j = 1, \ldots, n$$

$$x_{ij} = 0 \text{ or } 1, \quad i, j = 1, \ldots, n$$

- A 0-1 integer programming problem.

Outline

General formulation for constrained nonlinear optimization

- General form

$$\min f(\boldsymbol{x})$$

$$g_i(\boldsymbol{x}) \leq 0, \quad i = 1, 2, \ldots, m$$

$$h_j(\boldsymbol{x}) = 0, \quad j = 1, 2, \ldots, p$$

$$\boldsymbol{x} \in X \subset \mathbb{R}^n, \boldsymbol{x} = (x_1, x_2, \ldots, x_n)$$

and call the set

$$S = \left\{ \boldsymbol{x} | g_i(\boldsymbol{x}) \leq 0,\ i = 1, 2, \ldots, m; h_j(\boldsymbol{x}) = 0,\ j = 1, 2, \ldots, p; \quad \boldsymbol{x} \in X \right\}$$

the feasible solution of the problem.

Penalty method

- The idea of penalty method is to convert the constrained optimization problem into an unconstrained optimization problem by introducing a penalty term.
- Define the penalty function

$$F(\boldsymbol{x}, M) = f(\boldsymbol{x}) + Mp(\boldsymbol{x})$$

$M > 0$ is called penalty factor, $p(\boldsymbol{x})$ is called penalty term. In general $p(\boldsymbol{x}) \geq 0$ for arbitrary $\boldsymbol{x} \in \mathbb{R}^n$ and $p(\boldsymbol{x}) = 0$ iff $\boldsymbol{x} \in S$.

Penalty method

- For equality constrain define

$$g_j^+(\boldsymbol{x}) = (h_j(\boldsymbol{x}))^2, \quad j = 1, 2, \ldots, p$$

  and for inequality constrain define

$$g_{i+p}^+(\boldsymbol{x}) = \begin{cases} 0, & g_i(\boldsymbol{x}) \leq 0 \\ (g_i(\boldsymbol{x}))^2, & g_i(\boldsymbol{x}) > 0 \end{cases}$$

  for i=1,2,. . . ,m.

- Define $L = p + m$ and the penalty function

$$F(\boldsymbol{x}, M_k) = f(\boldsymbol{x}) + M_k \sum_{i=1}^{L} g_i^+(\boldsymbol{x})$$

  where $M_k > 0$ and

$$M_1 < M_2 < \cdots < M_k < \cdots \to +\infty$$

Penalty method

- If $M_k \gg 1$ and if the penalty function

$$F(\boldsymbol{x}, M_k) = f(\boldsymbol{x}) + M_k p(\boldsymbol{x})$$

  is minimized, this will force the penalty term

$$p(\boldsymbol{x}) \approx 0.$$

  Otherwise M_k will amplify many times!! That's why it is called penalty method.

- In general, take

$$M_{k+1} = cM_k, \quad c \in [4, 50]$$

Algorithm for penalty method

1. Take $M_1 > 0$, tolerance $\epsilon > 0$, Initial sate $\boldsymbol{x}_0$, set $k = 1$;
2. Solve the unconstrained optimization

$$\min F(\boldsymbol{x}, M_k) = f(\boldsymbol{x}) + M_k \sum_{i=1}^{L} g_i^+(\boldsymbol{x})$$

   with initial data $\boldsymbol{x}_{k-1}$, and the solution is $\boldsymbol{x}_k$;
3. Define

$$\tau_1 = \max\{|h_i(\boldsymbol{x}_k)|\}, \quad \tau_2 = \max\{g_i(\boldsymbol{x}_k)\}$$

   and $\tau = \max\{\tau_1, \tau_2\}$;
4. If $\tau < \epsilon$, over; otherwise, set $M_{k+1} = cM_k$, $k = k + 1$, return to step 2.

Barrier method

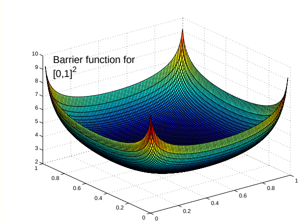
- Barrier method is suitable for optimization as

$$\min f(\boldsymbol{x}), \quad \text{s.t. } \boldsymbol{x} \in S$$

where S is a set characterized only by inequality constraints

$$S = \left\{ \boldsymbol{x} | g_i(\boldsymbol{x}) \leq 0, \; i = 1, 2, \ldots, m \right\}$$

- Graphical interpretation of barrier method



Barrier method

- Define barrier term $B(\boldsymbol{x})$ such that

$$B(\boldsymbol{x}) \geq 0 \ \text{ and } \ B(\boldsymbol{x}) \to \infty \ \text{ as } \ \boldsymbol{x} \to \text{boundary of } S$$

- Inverse barrier term

$$B(\boldsymbol{x}) = \sum_{i=1}^{m} g_i^+(\boldsymbol{x})$$

  and

$$g_i^+(\boldsymbol{x}) = -\frac{1}{g_i(\boldsymbol{x})}$$

- Logarithmic barrier term

$$g_i^+(\boldsymbol{x}) = -\ln(-g_i(\boldsymbol{x}))$$

- Barrier function

$$F(\boldsymbol{x}, r_k) = f(\boldsymbol{x}) + r_k B(\boldsymbol{x})$$

  where

$$r_k > 0, \quad r_1 > r_2 > \cdots > r_k > \cdots \to 0.$$

Barrier method

- Though the formulation of barrier method

$$F(\boldsymbol{x}, r) = f(\boldsymbol{x}) + rB(\boldsymbol{x}), \boldsymbol{x} \in S$$

is still a constrained optimization, but the property

$$F(\boldsymbol{x}, r) \to \infty \quad \text{as} \quad \boldsymbol{x} \to \text{boundary of } S$$

makes the numerical implementation an unconstrained problem.

- The implementation will be an iteration ($c \in [4, 10]$)

$$r_{k+1} = r_k / c$$

until some type of convergence criterion is satisfied.

Outline

Examples

Constrained optimization

Integer programming

Discrete optimization

- Integer programming is a typical case in discrete optimization. There are large amount of discrete optimization problems in graph theory and computer science.
- Discrete optimization models are, except for some special cases, are extremely hard to solve in practice. They are NP-Hard problem. (Is NP=P? This is a million dollar problem.)
- Unfortunately there are no general widely applicable methods for solving discrete problems. But there are some common themes such as relaxation, branch-and-bound etc.
- There are some heuristic ideas such as local search methods, simulated annealing, genetic algorithms etc.

Integer linear programming

- General form

$$\max z = \boldsymbol{c}^T \boldsymbol{x}$$

$$\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \quad \boldsymbol{x} \geq \boldsymbol{0}, \ x_i \in I, \ i \in J \subset \{1, 2, \cdots, n\}$$

where

$$\boldsymbol{x} = (x_1, x_2, \cdots, x_n), \boldsymbol{c} = (c_1, c_2, \cdots, c_n)$$

$$\boldsymbol{b} = (b_1, b_2, \cdots, b_m), \quad \boldsymbol{A} = (a_{ij})_{m \times n}, \quad I = \{0, 1, 2, \ldots\}$$

If $J = \{1, 2, \cdots, n\}$, it is a pure integer programming. If $J \neq \{1, 2, \cdots, n\}$, it is a mixed integer programming problem.

Relaxation and decomposition

- Relaxation: the problem obtained after relaxing some constrained condition is called relaxation problem of the primitive problem. For example we obtain the linear programming after relaxing the integer constraints.
- Decomposition: define $R(P)$ the feasible solution set of problem (P). If

$$\cup_{i=1}^{m} R(P_i) = R(P)$$

$$R(P_i) \cap R(P_j) = \emptyset \quad (1 \leq i \neq j \leq m),$$

we call the subproblems $(P_1), (P_2), \cdots, (P_m)$ a decomposition of (P).

Example

- Example

$$\max z = 5x_1 + 8x_2$$

$$x_1 + x_2 \leq 6, \quad 5x_1 + 9x_2 \leq 45$$

$$x_1, x_2 \in I = \mathbb{N} \cup \{0\}$$

- Relaxation: let $x_1, x_2 \geq 0$, it is a linear programming problem, the optimum is $\boldsymbol{x} = (2.25, 3.75)$ which does NOT belong to I!
- Decomposition: decompose the range of x_2 into

$$x_2 \geq 4 \text{ or } x_2 \leq 3.$$

We obtain two subproblems.

Branch-and-bound

- The basic framework of Branch-and-bound method is as follows
  1. Upper Bounds: Efficient methods for determining a good upper bound $UB(P)$;
  2. Branching Rules: Methods for replacing an instance (P) of the discrete optimization problem with some further "smaller" subproblems (P_l) such that some optimal solution of (P) maps to an optimal solution of a subproblem (P_l).
  3. Lower Bounds: Efficient heuristics that attempt to determine a feasible candidate solution S with as low a value as is practical, yielding the lower bound $LB(P)$.

Some definitions

- Define the floor and ceil function for any $a \in \mathbb{R}$

$$\lfloor a \rfloor := \text{The integer nearest to } a \text{ but less than } a$$

$$\lceil a \rceil := \text{The integer nearest to } a \text{ but bigger than } a$$

It's clear that

$$0 \leq a - \lfloor a \rfloor < 1, \quad 0 \leq \lceil a \rceil - a < 1$$

- Examples

$$\lfloor -\frac{1}{7} \rfloor = -1, \quad \lfloor \frac{1}{28} \rfloor = 0, \quad \lfloor \frac{7}{4} \rfloor = 1$$

$$\lceil -\frac{1}{7} \rceil = 0, \quad \lceil \frac{1}{28} \rceil = 1, \quad \lceil \frac{7}{4} \rceil = 2$$

Branching rules

- Define the optimal solution of the linear-programming relaxation as

$$\boldsymbol{x}^* = (x_1, x_2, \cdots, x_n)$$

- Branching rule: We choose a variable $x_k^* \notin \mathbb{Z}$. We branch by creating two new subproblems:
  1. (P') together with the additional inequality

$$x_k \leq \lfloor x_k^* \rfloor$$

  2. (P') together with the additional inequality

$$x_k \geq \lceil x_k^* \rceil$$

Branch-and-bound: an example

- Example

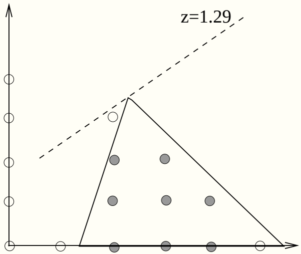
$$\max z = -x_1 + x_2$$

Subject to

$$12x_1 + 11x_2 \leq 63$$

$$-22x_1 + 4x_2 \leq -33$$

$$x_1, x_2 \geq 0, \quad x_1, x_2 \in \mathbb{Z}$$



Branch-and-bound: an example

- First solve the relaxation problem we have

Subprogram	z^*	x_1^*	x_2^*
IP	1.29	2.12	3.41

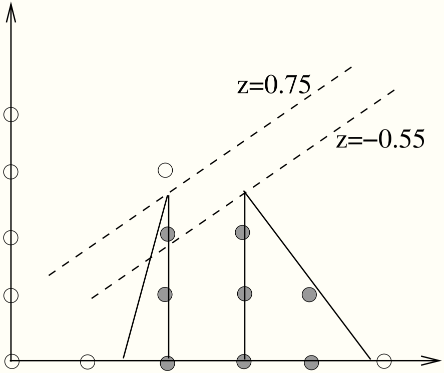
Then the lower and upper bounds

$$LB = -\infty, \quad UB = 1.29$$

- Branching x_1 we have two subprograms and solve the relaxation problems respectively

Subprogram	z^*	x_1^*	x_2^*
IP with $x_1 \leq 2$	0.75	2.00	2.75
IP with $x_1 \geq 3$	-0.55	3.00	2.45

Branch-and-bound: an example



Branch-and-bound: an example

- Branching x_2 of IP with $x_1 \leq 2$ and solve the relaxation problem

Subprogram	z^*	x_1^*	x_2^*
IP with $x_1 \leq 2, x_2 \leq 2$	0.14	1.86	2.00
IP with $x_1 \leq 2, x_2 \geq 3$	-	-	-

- Thus we have subprograms

Subprogram	z^*	x_1^*	x_2^*
IP with $x_1 \geq 3$	-0.55	3.00	2.45
IP with $x_1 \leq 2, x_2 \leq 2$	0.14	1.86	2.00

and

$$LB = -\infty, \quad UB = 0.14$$

Branch-and-bound: an example

- Branching x_1 of IP with $x_1 \leq 2, x_2 \leq 2$ we have

Subprogram	z^*	x_1^*	x_2^*
IP with $x_1 \leq 2, x_2 \leq 2, x_1 \leq 1$	-	-	-
IP with $x_1 \leq 2, x_2 \leq 2, x_1 \geq 2$	0.00	2.00	2.00

and because $\boldsymbol{x}^* \in \mathbb{Z}$ in the subprogram, we have

$$LB = 0.00, \quad UB = 0.14$$

- Because $-0.55 < LB = 0.00$, the subprogram

$$\text{IP with } \ x_1 \geq 3$$

is deleted.

- So finally we have the optimal solution

$$\boldsymbol{x}^* = (2, 2), \quad z^* = 0.00$$

Cutting-plane method

- A cutting-plane is a linear inequality that is generated as needed in the course of solving an integer linear program as a sequence of linear programs.
- Generic cutting-plane method
  1. Initially let LP be the linear programming relaxation of IP;
  2. Let $\boldsymbol{x}^*$ be an optimal extreme-point solution of LP;
  3. If $\boldsymbol{x}^*$ is all integer, then stop because $\boldsymbol{x}^*$ is optimal to IP;
  4. If $\boldsymbol{x}^*$ is not all integer, then find an inequality that is satisfied by all feasible solutions of IP, but is violated by $\boldsymbol{x}^*$, append the inequality to LP, and go to step 2.

Gomory cutting-plane

- For an equality constraints

$$x_1 + (-\frac{1}{7})x_3 + \frac{1}{28}x_4 = \frac{7}{4}$$

Perform transformation

$$\lfloor 1 \rfloor x_1 + \lfloor -\frac{1}{7} \rfloor x_3 + \lfloor \frac{1}{28} \rfloor x_4 - \lfloor \frac{7}{4} \rfloor = (\lfloor 1 \rfloor - 1)x_1$$

$$+(\lfloor -\frac{1}{7} \rfloor + \frac{1}{7})x_3 + (\lfloor \frac{1}{28} \rfloor - \frac{1}{28})x_4 + \frac{7}{4} - \lfloor \frac{7}{4} \rfloor$$

Gomory cutting-plane

- We have

$$x_1 - x_3 - 1 = -\frac{6}{7}x_3 - \frac{1}{28}x_4 + \frac{3}{4}$$

- Because

  1. x_1, x_3 are integers from the left hand side;
  2. $x_3, x_4 \in \mathbb{N} \cup \{0\}$ from the righthand side;

  we have the cutting plane

$$-\frac{6}{7}x_3 - \frac{1}{28}x_4 + \frac{3}{4} \leq 0$$

  or equivalently

$$x_1 - x_3 - 1 \leq 0$$

- Generating the inequality from the lower floor decomposition technique is called Gomory cutting plane method.

Concrete example of Gomory cutting plane method

- Example

$$\min z = -x_1 - 27x_2$$

$$-x_1 + x_2 \leq 1$$

$$24x_1 + 4x_2 \leq 25$$

$$x_1, x_2 \geq 0, x_1, x_2 \in I$$

- Transform into standard form and make relaxation

$$\min z = -x_1 - 27x_2$$

$$-x_1 + x_2 + x_3 \qquad = 1$$

$$24x_1 + 4x_2 \qquad + x_4 = 25$$

$$x_1, x_2 \geq 0$$

Concrete example of Gomory cutting plane method

- Simplex method for optimal solution

Basis	$\boldsymbol{a}_1$	$\boldsymbol{a}_2$	$\boldsymbol{a}_3$	$\boldsymbol{a}_4$	$\boldsymbol{b}$
$\boldsymbol{a}_2$	0	1	$\frac{6}{7}$	$\frac{1}{28}$	$\frac{7}{4}$
$\boldsymbol{a}_1$	1	0	$-\frac{1}{7}$	$\frac{1}{28}$	$\frac{3}{4}$

i.e. we have

$$\boldsymbol{x}^* = (\frac{3}{4}, \frac{7}{4})$$

- $\boldsymbol{x}^* \notin \mathbb{Z}$, we determine the cutting plane

$$\frac{3}{4} - \frac{6}{7}x_3 - \frac{1}{28}x_4 \leq 0$$

Transform into standard form we have

$$-24x_3 - x_4 + x_5 = -21$$

And supplement this constraint into the primitive constraints.

Concrete example of Gomory cutting plane method

- Simplex method for optimal solution

Basis	$\boldsymbol{a}_1$	$\boldsymbol{a}_2$	$\boldsymbol{a}_3$	$\boldsymbol{a}_4$	$\boldsymbol{a}_5$	$\boldsymbol{b}$
$\boldsymbol{a}_2$	0	1	0	0	$\frac{1}{28}$	1
$\boldsymbol{a}_1$	1	0	0	$\frac{1}{24}$	$-\frac{1}{168}$	$\frac{7}{8}$
$\boldsymbol{a}_3$	0	0	1	$\frac{1}{24}$	$-\frac{1}{24}$	$\frac{7}{8}$

i.e. we have

$$\boldsymbol{x}^* = (\frac{7}{8}, 1)$$

- $\boldsymbol{x}^* \notin \mathbb{Z}$, we determine the cutting plane

$$\frac{7}{8} - \frac{1}{24}x_4 - \frac{23}{24}x_5 \leq 0$$

Transform into standard form we have

$$-x_4 - 23x_5 + x_6 = -21$$

And supplement this constraint into the primitive constraints.

Concrete example of Gomory cutting plane method

- Simplex method for optimal solution

Basis	$\boldsymbol{a}_1$	$\boldsymbol{a}_2$	$\boldsymbol{a}_3$	$\boldsymbol{a}_4$	$\boldsymbol{a}_5$	$\boldsymbol{a}_6$	$\boldsymbol{b}$
$\boldsymbol{a}_2$	0	1	0	0	$\frac{1}{28}$	0	1
$\boldsymbol{a}_1$	1	0	0	0	$-\frac{27}{28}$	$\frac{1}{24}$	0
$\boldsymbol{a}_3$	0	0	1	0	-1	$\frac{1}{24}$	0
$\boldsymbol{a}_4$	0	0	0	1	23	-1	21

i.e. we have

$$\boldsymbol{x}^* = (0, 1)$$

- $\boldsymbol{x}^* \in \mathbb{Z}$, so we obtain the optimal solution

$$\boldsymbol{x}^* = (0, 1), \quad z^* = -27$$

The geometric meaning of Gomory cutting plane

Transforming the cutting plane into planes with primitive varibales x_1, x_2, we have the cutiting plane equations

$$\text{Cutting plane 1} \quad x_2 \leq 1$$

$$\text{Cutting plane 2} \quad x_1 + 27x_2 \leq 27$$