Lecture 25

§1 Review of the branch and bound method

1. Branching procedure

Branching Procedures:

1. Solve the LP relaxation
   - If the optimal solution is integral, then it is optimal to IP
   - Otherwise go to step 2
2. If the optimal solution to the LP relaxation is $\mathbf{x}^*$ and x_i^* is fractional, then branch the problem into the following two:
   - 2.1 One with an added constraint that $x_i \le \lfloor x_i^* \rfloor$
   - 2.2 One with an added constraint that $x_i \ge \lceil x_i^* \rceil$
3. For each of the two problems, use the same method to solve them, and get optimal solution $\mathbf{x}_1^*$ and $\mathbf{x}_2^*$ with optimal value v_1^* and v_2^*
   - Compare and get the optimal solution

2. Bounding procedure

Bounding procedures (for maximization):

- Any LP relaxation solution can provide an upper bound for each node in the branching process
- Any feasible solution to the IP can provide a lower bound for the entire problem
- We call the best integer solution obtained for the problem the "incumbent solution"

When at a certain node, the optimal value of the LP relaxation of this branch is even less than the current lower bound (the objective value of the incumbent solution). Then we should abandon this branch (also called prune or fathom that branch)

- No better solution can be obtained from exploring this branch

Bounding is very important for branch-and-bound, it is the key to make it efficient (and practical)

§2 Branch selection

我们需要决定是 go deep into one branch first 还是 go wide to solve all problems on a given level first

1. Go deep / go wide

在 branch-and-bound algorithm 中，go deep 更好（效率更高）

- 多数 integral solutions 均在 tree 的深处，先得到 integral feasible solutions 可在后续 bounding procedure 中使用
- Memory efficient，由 parent node 得到的 LP 仅增加了一个 constraint，可使用 sensitivity analysis
- Coding 更容易（recursion）

2. Branch 的选取

当有两个 branches 可选时，我们希望选取"离 optimal solution"更近的 branch

- 没有 work-for-all theory 来帮助选取
- 一个 heuristic method 是选取 LP relaxation optimal value 更优的 branch
- 目前在尝试用 ML 辅助选取

3、Branch-and-bound method 的 complexity

1° Branch-and-bound method 为一个 enumeration method

- 最坏情况下需要遍历 region 内的所有 feasible integral solutions (exponential in the problem size)
- NP-hard

2° 但通常情况下，仅需遍历一小部分 feasible integral solutions

§3 Binary one-constraint linear optimization

1. LP relaxation 的问题形式

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} c_i x_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} a_i x_i \le C \qquad (a_i, c_i > 0) \\ & 0 \le x_i \le 1 \end{aligned}$$

2. 该 LP relaxation 的简便解法

考虑 value-to-weight $\frac{c_i}{a_i}$（单位代价带来的收益），则

Optimal solution 使得 $\frac{c_i}{a_i}$ 越高的 x_i 值越高（上限为1），且 $\sum_{i=1}^{n} a_i x_i = C$

证明：

W.T.S. 存在一个 threshold η，使得

$$\begin{cases} \text{若 } \frac{c_i}{a_i} > \eta \Rightarrow x_i = 1 \\ \text{若 } \frac{c_i}{a_i} < \eta \Rightarrow x_i = 0 \\ \text{若 } \frac{c_i}{a_i} = \eta \Rightarrow x_i \in [0,1] \quad (\text{使得 } \sum_{i=1}^{n} a_i x_i = C) \end{cases}$$

- 首先求出 LP relaxation 的 dual problem

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} c_i x_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} a_i x_i \le C \quad \text{---} \ y \\ & x_i \le 1 \quad \text{---} \ \lambda_i \\ & x_i \ge 0 \end{aligned}$$

dual problem 为

$$\begin{aligned} \min_{y, \lambda_i} \quad & Cy + \sum_{i=1}^{n} \lambda_i \\ \text{s.t.} \quad & a_i y + \lambda_i \ge c_i \quad \forall i = 1, \dots, n \quad \text{----} \ x_i \\ & y, \lambda_i \ge 0 \quad \forall i = 1, \dots, n \end{aligned}$$

- 注意到 $\lambda_i \ge c_i - a_i y$ 且 $\lambda_i \ge 0$，因此若我们 fix y，则为了 minimize $Cy + \sum_{i=1}^{n} \lambda_i$，一定有

$$\begin{aligned} \lambda_i &= \max\{c_i - a_i y, 0\} \\ &= a_i \max\{\tfrac{c_i}{a_i} - y, 0\} \\ &= \begin{cases} c_i - a_i y & \text{若 } \frac{c_i}{a_i} > y \\ 0 & \text{若 } \frac{c_i}{a_i} < y \end{cases} \end{aligned}$$

- 由互补松弛性可知，

$$\begin{cases} \lambda_i^* (1 - x_i^*) = 0 \\ x_i^* (\lambda_i^* - (c_i - a_i y^*)) = 0 \end{cases}$$

因此我们有

① 若 $\frac{c_i}{a_i} > y^*$ $\Rightarrow$ $\lambda_i^* = c_i - a_i y^*$ $\Rightarrow$ $\lambda_i^* \neq 0$ $\Rightarrow$ $x_i^* = 1$

② 若 $\frac{c_i}{a_i} < y^*$ $\Rightarrow$ $\lambda_i^* = 0$ $\Rightarrow$ $\lambda_i^* - (c_i - a_i y^*) \neq 0$ $\Rightarrow$ $x_i^* = 0$

取 $\eta = y^*$ 即可证出该结论

Q.E.D.

3. Binary one-constraint linear optimization

对于此类问题，可以使用 value-to-weight ratio 求解 LP relaxation，直到出现 infeasible subproblem

注：对于 binary problem，branching 后添加的 constraints 分别为 $x_i = 0$ 与 $x_i = 1$

例：一个 binary linear program：Knapsack problem

$$\begin{aligned} \text{maximize} \quad & 8x_1 + 11x_2 + 6x_3 + 4x_4 \\ \text{subject to} \quad & 5x_1 + 7x_2 + 4x_3 + 3x_4 \le 14 \\ & x_j \in \{0, 1\}, \quad j = 1, \cdots, 4 \end{aligned}$$

① 求出 value-to-weight

$\frac{c_1}{a_1} = \frac{8}{5}$, $\frac{c_2}{a_2} = \frac{11}{7}$, $\frac{c_3}{a_3} = \frac{3}{2}$, $\frac{c_4}{a_4} = \frac{4}{3}$

$\frac{c_1}{a_1} > \frac{c_2}{a_2} > \frac{c_3}{a_3} > \frac{c_4}{a_4}$

② 求解 LP relaxation

$x_1^* = 1$, $x_2^* = 1$, $x_3^* = 0.5$, $x_4^* = 0$, $z^* = 22$

③ Branching for x_3

(S_1): one with an additional constraint $x_3 = 0$

(S_2): one with an additional constraint $x_3 = 1$

求解 (S_1) 与 (S_2) 的 LP relaxation:

(S_1): $x_1^* = 1$, $x_2^* = 1$, $x_3^* = 0$, $x_4^* = \frac{2}{3}$, $z^* = 21.65$

(S_2): $x_1^* = 1$, $x_2^* = \frac{5}{7}$, $x_3^* = 1$, $x_4^* = 0$, $z^* = 21.85$

由于 (S_2) 的 z^* 更优，我们选取 (S_2) 进行 branching

④ Branching for x_2

(S_3): one with an additional constraint $x_2 = 0$

(S_4): one with an additional constraint $x_2 = 1$

求解 (S_3) 与 (S_4) 的 LP relaxation:

(S_3): $x_1^* = 1$, $x_2^* = 0$, $x_3^* = 1$, $x_4^* = 1$, $z^* = 18$ (integral)

(S_4): $x_1^* = 0.6$, $x_2^* = 1$, $x_3^* = 1$, $x_4^* = 0$, $z^* = 21.8$

⑤ Branching for x_1

(S_5): one with an additional constraint $x_1 = 0$

(S_6): one with an additional constraint $x_1 = 1$

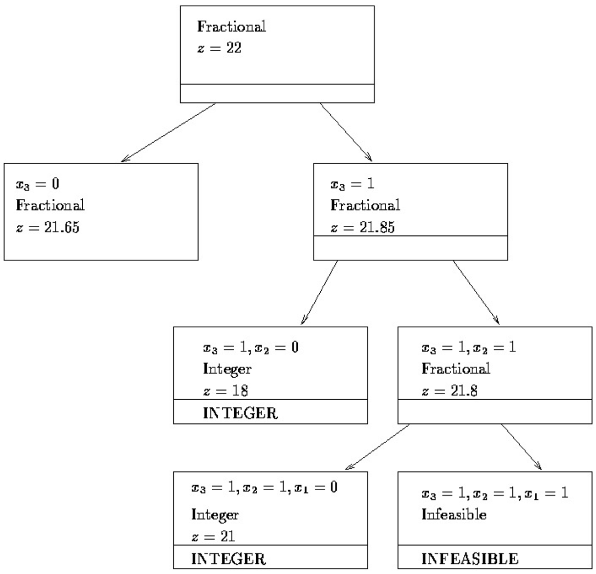
求解 (S_5) 与 (S_6) 的 LP relaxation:

(S_5): $x_1^* = 0$, $x_2^* = 1$, $x_3^* = 1$, $x_4^* = 1$, $z^* = 21$ (integral)

(S_6): infeasible

⑥ 由于 (S_2) 的 LP relaxation optimal value 为 21.65, 故 (S_2) 的最优值至多为 21

而 (S_5) 得到的 lower bound 为 21, 因此原问题的 optimal value 为 21, optimal solution 为 $(0,1,1,1)$



There are 16 possible combinations in total, but we don't need to visit all of them

- Bounding is very important, it can greatly reduce the search space
- In the above example, we don't need to consider the $x_3 = 0$ branch because of bounding