RMC

Lesson 17. Branch-and-bound

1 Today...

- "Combinatorial Explosion" of IPs
- Branch-and-bound for solving IPs

2 "Combinatorial Explosion" of IPs

Consider the knapsack problem (with all nonnegative parameters) in which we choose the most valuable collection of items to fit into a limited size "knapsack":

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{n} c_i x_i \\ \text{subject to} \quad & \sum_{i=1}^{n} a_i x_i \leq b \\ & x_i \in \{0,1\}, \text{ for } i = 1, 2, \ldots, n, \end{aligned}$$

- decision variable $x_i =$ $\boxed{1}$ if we choose to pack item i, $x_i =$ $\boxed{0}$ otherwise;
- c_i represents the $\boxed{\text{value}}$ of item i;
- a_i represents the $\boxed{\text{size}}$ of item i;
- b represents the $\boxed{\text{Capacity}}$ of the knapsack.

How many possible Solutions exist?

2^5

Problem 1. Suppose $n = 5$.

(not necessarily feasible)

(a) Write one possible solution to the knapsack problem. $(x_1, x_2, x_3, x_4, x_5) =$ $\boxed{(0,0,0,0,0)}$

(b) Which items does your solution recommend that you pack?

None!

(c) If the constraint eliminates half of the possible solutions, how many feasible solutions are there?

$2^5/2 = 2^4 = 16$

> **Complete enumeration** is a solution strategy for the knapsack problem (or any bounded IP) in which
>
> - the objective value is computed for *every feasible solution*;
> - the solution with the maximum objective function value is chosen as the optimal solution.

Now suppose $n = 100$ (a moderately-sized problem) and that the knapsack constraint eliminates half of the possible solutions. In a **complete enumeration** strategy in which we can check *one billion solutions per second*,

- there are

$$2^{100}/2 \quad \boxed{2^{99}} \approx 6.3 \times 10^{29} \quad \frac{6.3 \times 10^{29}}{1{,}000{,}000{,}000}$$

  feasible solutions to check,

- requiring about

$$6.3 \times 10^{20} \text{ seconds} \quad \approx \quad \boxed{2 \times 10^{13}} \quad \text{years}$$

  for complete enumeration.

> In general, for even moderately-sized problems, **complete enumeration** is a totally (AWESOME or HOPELESS) solution strategy.

And this is why we have branch-and-bound...

3 Branch-and-bound for solving IPs

> **Branch-and-bound** is an algorithm for solving mixed-integer programs:
>
> - the feasible region is iteratively subdivided to create smaller subproblems ("branching" phase);
> - the subproblems are bounded by solving relaxations ("bound" phase).

Typically, modern IP (or MIP) solvers use some variation of branch-and-bound.

3.1 Branching a subproblem on a variable

To **branch** means to split a problem into two smaller subproblems.

- For example, to find the tallest midshipman in the brigade:
  - solve subproblem 1: Find tallest mid in First Reg
  - solve subproblem 2: Find tallest mid in Second Reg
  - compare these two solutions.
- The union of the feasible regions (FRs) of the subproblems should be equivalent to the FR of the original problem.
- For example, consider the IP below:

$$\begin{aligned} (P1) \quad z^*_{IP} = \max\ & 8x + 7y \\ \text{s.t.} \quad & -18x + 38y \le 133 \\ & 13x + 11y \le 125 \\ & 10x - 8y \le 55 \\ & x, y \in \mathbb{Z}^{\ge 0} \end{aligned}$$

  Find an upper bound for z^*_{IP} by solving the LP relaxation of (P1). Suppose the LP relaxation has optimal solution $(x,\ y) = (4.75,\ 5.75)$ with optimal objective value $z^*_{LP} = 78.25$. This provides the following bound on z^*_{IP}:

$$z^*_{IP} \le \boxed{78} = \lfloor 78.25 \rfloor$$

  - Since the optimal LP solution, $(4.75, 5.75)$, is not integer-valued, we must **branch on one of the fractional-valued variables**. Let's choose to **branch on** x.
  - We know that x must be integer-valued, so we can eliminate all the fractional values between 4 and 5. Our two subproblems leverage this fact:

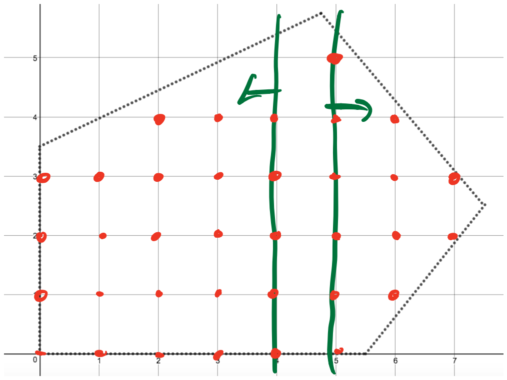
x must no more than 4 or...

$$\begin{aligned} (P2) \quad z^*_{IP} = \max\ & 8x + 7y \\ \text{s.t.} \quad & -18x + 38y \le 133 \\ & 13x + 11y \le 125 \\ & 10x - 8y \le 55 \\ & \boxed{x \le 4} \\ & x, y \in \mathbb{Z}^{\ge 0} \end{aligned}$$

at least 5.

$$\begin{aligned} (P3) \quad z^*_{IP} = \max\ & 8x + 7y \\ \text{s.t.} \quad & -18x + 38y \le 133 \\ & 13x + 11y \le 125 \\ & 10x - 8y \le 55 \\ & \boxed{x \ge 5} \\ & x, y \in \mathbb{Z}^{\ge 0} \end{aligned}$$

- Continuing the previous example, the *relaxed* feasible region of (P1) is shown below. Sketch the *relaxed* feasible regions of (P2) and (P3):



- In order to make progress in branch-and-bound:

  i.e. we eliminate some fractional solutions to the LP Relaxation

  The union of the *relaxed* FRs of the subproblems should be a PROPER subset of the *relaxed* FR of the original problem. (I.e., we *tighten* our formulation as we go.)

3.2 Branch-and-bound terminology

- As we branch on integer variables, we keep track of progress in a **branch-and-bound** Tree . **Nodes** represent Subproblems to the LP Relaxation .
- At some point during the algorithm, we will encounter an *integer feasible* solution which becomes the **incumbent solution**.
  - The incumbent solution provides a (LOWER / UPPER) bound on the global maximum.
  - If later we find a (BETTER / WORSE) feasible solution, it becomes the new incumbent solution.

  > The **incumbent solution** is the BEST feasible solution found so far.

- To **fathom** (or **prune**) a node means to eliminate its subproblem from consideration.
  - We know this part of the feasible region (CONTAINS / DOES NOT CONTAIN) ~~the~~ an optimal solution.)
- Leaf nodes are (BRANCHED / UNBRANCHED) nodes.

  There are 3 types of **leaf nodes**.

  (1) Fathomed

  (2) contains **current** Incumbent **solution** (only one of these nodes!)

  (3) **active** – still requires branching

3.3 Algorithm

Branch-and-bound for solving IPs

(Initialize)

- The root node (original problem) is the only active node.
- Set global lower bound: $\underline{z} = -\infty$. There is no incumbent solution $\underline{x}$ to start.

(Iterate)

- Select an **active** node. Branch on a **fractional** variable to create two subproblems.
- For each subproblem (SP):

  - Solve its relaxation (LP), if possible, for optimal solution x with objective value z:

    (LP) infeasible $\Rightarrow$ **fathom** (SP).

    $z \leq \underline{z} \Rightarrow$ **fathom** (SP).

    $z > \underline{z}$:

    x integer $\Rightarrow$ x becomes new **incumbent solution**: $\underline{x} = x$, $\underline{z} = z$.
    Fathom nodes whose upper bounds are less than the new $\underline{z}$.

    x fractional $\Rightarrow$ (SP) becomes **active** with local upperbound $\lfloor z \rfloor$.

(Stopping condition)

- When there are no more active nodes, the incumbent solution is the optimal solution to the original problem.

MIP gap

Another common stopping criterion is a predetermined "MIP gap", which is a measure of how close the current global lower bound, LB, and global upper bound, UB, are, as a fraction of the LB:

$$MIP\ gap = \boxed{\frac{|UB - LB|}{|LB|}}$$

Current obj. value $(\underline{z})$ corresponding to incumbent solution

When the MIP gap is small, the current incumbent solution is guaranteed to be *close* to optimal. For hard problems, we can tell the solver to stop at a "good" solution by setting a MIP gap flag. In Pyomo, this looks like:

```
pyo.SolverFactory('glpk').solve(model, mipgap=0.01 )
```

Branch-and-bound Example

To demonstrate the branch-and-bound algorithm, we will work through an example using the following documents:

- L17_Branch_And_Bound_Example.pdf
- L17_Branch_And_Bound_Example.ipynb

Branching Rules

The procedure in the box on the previous page is really an *algorithmic framework*, rather than an actual algorithm. To become a true algorithm, we would need to specify

- a **branching rule** (to decide which $\boxed{\text{active}}$ node to branch);
- a **variable selection rule** (to decide which $\boxed{\text{fractional}}$ variable to branch on).

Possible branching rules:

- **depth-first search:** Quickly go deep in the tree by always branching on one of the most recently constructed active nodes, for example.
  - ADVANTAGE: We get an Integer feasible $\boxed{\text{Incumbent}}$ solution quickly, which we require in order to fathom feasible nodes.
  - DISADVANTAGE: Long computation times, if we keep diving down to feasible solutions in every successive subproblem.
- **best-first search:** Branch on the active node with the $\boxed{\text{best (largest)}}$ *local upper bound.*
  - ADVANTAGE: We explore promising regions early on.
  - DISADVANTAGE: It may take a long time to get a first incumbent solution. It also tends to produce a very wide tree, requiring a lot of memory to maintain many active nodes.
- a **hybrid approach** is most common. For example, use depth-first to get an incumbent solution. Switch to best-first to search promising nodes.

3.4 Branch-and-bound variations

In reality, modern MILP solvers build on the basic b-and-b framework presented here.

Most solvers attain a **lower-bounding feasible solution in every subproblem** by solving a *restriction* of the subproblem (rather than a relaxation). The best of these feasible solutions is maintained as the incumbent solution.

Most IP/†MIP solvers employ a variation of branch-and-bound called **branch-and-cut**. In some subproblems, new constraints are generated in a "cutting-plane" phase to tighten the formulation of the subproblem relaxation (LP). This results in

- a *less fractional* optimal solution x to the subproblem (LP);
- a *tighter upperbound* z on the subproblem;
- *fewer branching nodes* overall.