Lesson 14. Branch-and-bound

1 Brief Review

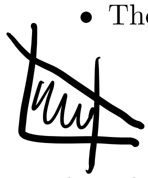
There are a few major things from the last few classes to remember:

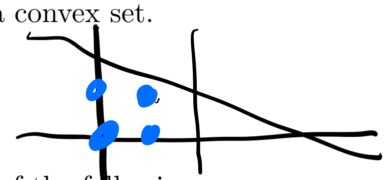
1. Definition of a convex set:

For any 2 points in set the line connecting those points is also in the set.

For all $x, y \in S$ $\lambda x + (1-\lambda) y \in S$ $\lambda \in [0,1]$

   - The feasible region of every (LP or IP) is a convex set.

LP 

IP 

2. The solution to every IP is **EXACTLY ONE** of the following:

   (a) Unique optimal Solution

   (b) 2 or more optimal Solutions

   (c) Infeasible

   (d) Unbounded

3. The relationship between the solutions of an IP and its LP relaxation.

> Let z^*_{IP} be the optimal objective function of the IP, let z^*_{LP} be the optimal objective function value of the LP, and let $\hat{z}$ be the objective function value associated with some **feasible solution** of the IP. Then:
>
> - If we are minimizing:
> $$z^*_{LP} \leq z^*_{IP} \leq \hat{z}$$
> - If we are maximizing:
> $$\hat{z} \leq z^*_{IP} \leq z^*_{LP}$$

This relationship is key to understanding the branch and bound algorithm!

2 "Combinatorial Explosion" of IPs

A straightforward way to solve IPs is to take advantage of the fact that the feasible region of an IP is a grid of integer points. Since this is the case, why not just check every point in the grid?

Consider the knapsack problem (with all nonegative parameters) in which we choose the most valuable collection of items to fit into a limited size "knapsack" (this is from week 1):

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

- decision variable $x_i =$ 1 if we choose to pack item i, $x_i =$ 0 otherwise;
- c_i represents the Usefulness of item i;
- a_i represents the weight of item i;
- b represents the Capacity of the knapsack.

Problem 1. Suppose $n = 5$.

(a) Write one possible solution to the knapsack problem. $(x_1, x_2, x_3, x_4, x_5) =$ 1, 1, 0, 0, 1

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

Packing 1, 2, 5
leaving 3, 4

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

How many possible solutions: $\frac{}{2}\ \frac{}{2}\ \frac{}{2}\ \frac{}{2}\ \frac{}{2} = 2^5 = 32$

If constraint elims half $2^5/2 = 2^4 = 16$

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*,

$n=100 \to 2^{100}$ Constraint elims half 2^{99}

- there are

$$\boxed{2^{99}} \approx 6.3 \times 10^{29}$$

Total # of solutions for 100 items

feasible solutions to check,

- requiring about

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

Pyomo → < 1 second

for complete enumeration. Total time if one billion solutions / per second

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: Tallest person in companies 1-15
  - solve subproblem 2: Tallest person in 16 - 30
  - compare these two solutions.
- The union of the feasible regions (FRs) of the subproblems should be equal the FR of the original problem.
- For example, consider the IP below:

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

LP Relaxation

$$
\begin{aligned}
z_{LP} = \max\ & 8x+7y \\
& -18x + 38y \le 133 \\
& 13x + 11y \le 125 \\
& 10x - 8y \le 55 \\
& x, y \ge 0
\end{aligned}
$$

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}:

$$z^*_{IP} \le \boxed{78}$$

Rounding 78.25

x=4.75 LP Not feasible for IP.

- 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.

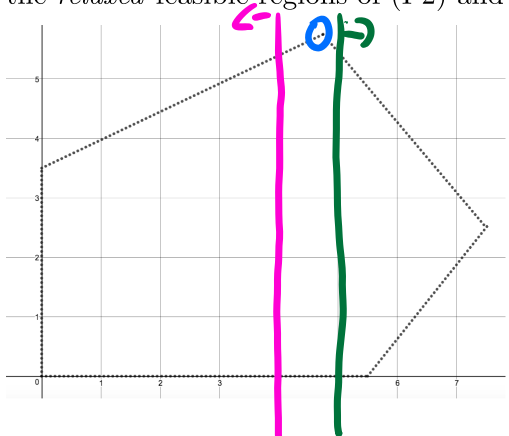
$$
\begin{aligned}
(P3) \quad z^*_{IP} = \max\ & 8x + 7y \\
\text{s.t.}\quad & -18x + 38y \le 133 \\
& 13x + 11y \le 125 \\
& 10x - 8y \le 55 \\
& \boxed{x \ge 5} \\
& x, y \in \mathbb{Z}^{\ge 0}
\end{aligned}
$$

P1
LP not feasible for IP.

$$P1 \xrightarrow{x \le 4} P2 \qquad P1 \xrightarrow{x \ge 5} P3$$

Same IP feasible region but x=4.75 not feasible

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



P1: $z^*_{LP} = 78.25$

$(x, y) = (4.75, 5.75)$

P2: original with $x \leq 4$

P3: original with $x \geq 5$

P2 and P3 together have same IP feasible region, but FR of LP has shrunk

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

  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.)

  ↑ LP formulation gets smaller

Tree

3.2 Branch-and-bound terminology

P1

P2 P3

- 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**. ← one problem at least will give an integer 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 optimal solution.)

- Leaf nodes are (BRANCHED / UNBRANCHED) nodes.

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

  (1) Pruned

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

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

optimal solution definitely not here →

Best you have now →

↑ Optimal solution may be in that region

P1 $z \leq 78$

P2 integer $z = 77$ ↑ Incumbent

P3 LP $z \leq 75$ ↑ Pruned

3.3 Algorithm

Branch-and-bound for solving IPs

(Initialize)

- The root node (original problem) is the only active node.] Original problem LP Relaxation
- 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): pick any fractional variable

  - 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).] fractional but $z \leq$ lower bound

    $z > \underline{z}$:] Solution fractional with $z >$ lower bound

    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 Optimal solution → No more Regions to explore

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

MIP Gap: How far from optimal

$$\text{MIP GAP} = \frac{\text{upper bound} - \text{lower bound}}{\text{lower bound}}$$

When the MIP gap is small, the current solution is *close* to the optimal solution. For hard to solve problems, we often may stop when the MIP gap gets small enough.

1% → stop when we found a solution that is within 1% of the best.

Branch-and-bound Example

To demonstrate the branch-and-bound algorithm, we will work through two examples using the following documents. The way we solve problems here is exactly what you should do for your homework!

- `Lesson 14_Branch_and_Bound_Graph-Example.pdf`] Friday
- `Lesson 14_Branch_And_Bound_Python-Example.pdf`] Monday
- `L14_Branch_And_Bound_Example.ipynb`] post but supplement to Monday

Branching Rules

The procedure in the box on page 6 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 $\boxed{\text{incumbent}}$ 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 $\boxed{\text{largest}}$ *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.