Enumerative Methods

A knapsack problem

- Let’s focus on maximization integer linear programs with only binary variables
- For example: a knapsack problem with 6 items

$$\text{max } 16x_1 + 22x_2 + 12x_3 + 8x_4 + 11x_5 + 19x_6$$
$$\text{s.t. } 5x_1 + 7x_2 + 4x_3 + 3x_4 + 4x_5 + 6x_6 \leq 14$$
$$x_1, x_2, \ldots, x_6 \in \{0, 1\}$$

Complete enumeration

- Complete enumeration systematically considers all possible solutions
  - $n$ binary variables $x_1, \ldots, x_n \Rightarrow 2^n$ possible solutions
- After considering all possible solutions, choose best feasible solution
- Usual idea: iteratively break the problem into 2
  - For example, first, we consider separately the cases that $x_1 = 0$ and $x_1 = 1$

An enumeration tree

- Let’s enumerate all possible solutions of our illustrative knapsack problem
- 6 binary decision variables $x_1, \ldots, x_6$
- We can enumerate all possible solutions systematically using a tree
- Start with root node
  - No variables have been fixed in value

- **Branch** the possibilities for $x_1$: $x_1 = 0$ or $x_1 = 1$
Next, branch the possibilities for $x_2$: $x_2 = 0$ or $x_2 = 1$

Keep building the tree, branching the possibilities for $x_3$, $x_4$, $x_5$, $x_6$

Each node corresponds to a partial solution
- For example, node 4 $\iff$ fix $x_1 = 0$ and $x_2 = 1$
- partial solution of node $4 = \mathbf{x}^{(4)} = (0, 1, #, #, #, #)$

Each of the 64 leaves of the tree (nodes at the bottom) corresponds to a solution: a complete assignment of variables

Subtrees of an enumeration tree

Subtree (or descendants) of node $i =$ nodes obtained from node $i$ from subsequent branching
• Example: red nodes = subtree of node 4
• Recall: node 4 ⇔ partial solution \( x^{(4)} = (0, 1, \#, \#, \#, \#) \)
• Leaves of subtree of node 4 ⇔ completions of \( x^{(4)} \)
  – (full) solutions that have the same fixed variables as \( x^{(4)} \)
• Idea: stop branching from a node as soon as possible
  – Suppose we look at node 4 and conclude none of its descendants can be optimal
    ⇒ Can eliminate 1/4 the solutions at once!

**Incumbent solutions**

• Goal of branch and bound: find an optimal (or at least a good feasible) solution to some optimization model
• The **incumbent solution** at any stage of branch and bound is the best feasible solution known so far (in terms of objective value)
• Notation:
  – Incumbent solution \( \hat{x} \)
  – Incumbent solution’s objective function value \( \hat{v} \)
• Most branch and bound algorithms have subroutines that run at the beginning trying to get a good feasible solution

**Eliminating nodes and subtrees**

• Let’s look at our knapsack problem
• Suppose that we have an incumbent solution \( \hat{x} \) with objective value \( \hat{v} \):
  \[
  \hat{x} = (1, 1, 0, 0, 0, 0) \quad \hat{v} = 38
  \]
• Let’s look at the subtree of node 4 in our enumeration tree
• Node 4 ⇔ partial solution $x^{(4)} = (0, 1, \#, \#, \#, \#)$

• All possible completions of $x^{(4)}$ ⇔ Leaves of node 4’s subtree

• **Candidate problem** for node 4: find the best possible completion of $x^{(4)}$

\[
v^{(4)} = \max \quad 16x_1 + 22x_2 + 12x_3 + 8x_4 + 11x_5 + 19x_6 \\
s.t. \quad 5x_1 + 7x_2 + 4x_3 + 3x_4 + 4x_5 + 6x_6 \leq 14 \\
x_1 = 0, x_2 = 1 \\
x_1, x_2, \ldots, x_6 \in \{0, 1\}
\]

• LP relaxation gives us upper bound on $v^{(4)}$:

\[
\tilde{v}^{(4)} = \max \quad 16x_1 + 22x_2 + 12x_3 + 8x_4 + 11x_5 + 19x_6 \\
s.t. \quad 5x_1 + 7x_2 + 4x_3 + 3x_4 + 4x_5 + 6x_6 \leq 14 \\
x_1 = 0, x_2 = 1 \\
0 \leq x_i \leq 1, \quad i = 1, \ldots, 6
\]

• Solve LP relaxation: $\tilde{v}^{(4)} = 44$

• Best completion of $x^{(4)}$ has value $v^{(4)} \leq \tilde{v}^{(4)} = 44$

• Incumbent solution has value $\hat{v} = 38$

$\Rightarrow$ It is possible that some completion of $x^{(4)}$ has a better solution value than 38

$\Rightarrow$ Need to examine solutions that branch from node 4

• What if we had an incumbent solution with value $\hat{v} = 45$?

• Then no completion of $x^{(4)}$ is better than our incumbent, since

\[
v^{(4)} \leq \hat{v}^{(4)} = 44 < 45 = \hat{v}
\]

• We can **terminate** or **fathom** node 4: we do not need to branch the subtree of node 4
Branch and bound in a nutshell

- Branch and bound creates the enumeration tree
  - one node at a time
  - one branch at a time

- Before branching on a node $j$, it solves the LP relaxation of the node $j$’s candidate problem
  - **Candidate problem**
    * original problem with variables fixed according to the partial solution $x^{(j)}$ corresponding to node $j$
    * finds best completion of partial solution $x^{(j)}$

- Depending on the solution to the candidate problem, it either
  - terminates node $j$
  - branches on node $j$

- We will examine 4 cases

Termination by infeasibility

- Node $j$ ↔ partial solution $x^{(j)}$
- Feasible region of candidate problem of node $j$ ↔ All possible completions of $x^{(j)}$ ↔ All leaves of node $j$’s subtree

**Case 1: Termination by infeasibility.** The LP relaxation of the candidate problem of node $j$ is infeasible

⇒ The candidate problem of node $j$ is infeasible

⇒ Any completion of the partial solution $x^{(j)}$ is infeasible for the original problem!

⇒ Terminate node $j$ (do not branch from node $j$)
Termination by bound

- Notation:

  \( \hat{v} = \) value of incumbent solution  
  \( v^{(j)} = \) optimal value of candidate problem for \( j \)  
  \( \bar{v}^{(j)} = \) optimal value of LP relaxation of candidate problem for \( j \)

- Recall: candidate problem of \( j \) finds best completion of partial solution \( x^{(j)} \)

- **Case 2: Termination by bound.** \( \bar{v}^{(j)} \leq \hat{v} \)

  \[ v^{(j)} \leq \bar{v}^{(j)} \leq \hat{v} \]

  \[ \Rightarrow \text{No completion of } x^{(j)} \text{ is better than the incumbent} \]

  \[ \Rightarrow \text{Terminate node } j \text{ (do not branch from node } j) \]

**Termination by solving**

- **Case 3: Termination by solving.** \( \bar{v}^{(j)} > \hat{v} \) and the optimal solution \( \bar{x}^{(j)} \) of the LP relaxation of node \( j \)'s candidate problem is integer

  - \( \bar{x}^{(j)} \) is integer \( \Rightarrow \bar{x}^{(j)} \) is optimal for the candidate problem

  \[ v^{(j)} = \bar{v}^{(j)} > \hat{v} \]

  \[ \Rightarrow \text{We have found a feasible solution that is better than the incumbent} \]

  \[ \Rightarrow \text{Save solution } x^{(j)} \text{ as new incumbent} \]
⇒ No completion of partial solution $x^{(j)}$ will be better
⇒ Terminate node $j$ (do not branch from node $j$)

**Branching**

- **Case 4: Branching.** $\bar{v}^{(j)} > \bar{v}$ and the optimal solution $\bar{x}^{(j)}$ of the LP relaxation of node $j$’s candidate problem is not integer

⇒ It is possible that a completion of the partial solution $x^{(j)}$ may have a better objective value

- Branch at node $j$: pick some variable that is not fixed in the partial solution $x^{(j)}$ and create a child node for each possible value

**Active nodes**

- A node is called **active** if it has been analyzed:
  - it has no children
  - it has not been terminated

- For example:

The active nodes here are 2 and 3

- Initially, the only active node is the root node 0
- Branch and bound stops when there are no more active nodes
LP-based branch and bound algorithm for 0-1 ILPS

- We have essentially described the whole branch and bound algorithm, piecemeal
- We’ll give an abbreviated version of the algorithm
- $A = \text{set of active nodes}$
- $\mathbf{x}$ = incumbent solution, $\hat{v}$ = value of incumbent solution
- $\text{LP}(t)$ = LP relaxation of node $t$’s candidate problem
- $\tilde{x}(t) = \text{optimal solution to } \text{LP}(t)$, $\tilde{v}(t) = \text{optimal value of } \text{LP}(t)$

0. Initialize.
- $A \leftarrow \{\text{partial solution with no variables fixed}\}$
- $\mathbf{x} \leftarrow \emptyset$, $\hat{v} \leftarrow -\infty$ (or some external heuristic finds an incumbent)
- Solution counter $t \leftarrow 0$

1. Select.
- If $A = \emptyset$, then $\mathbf{x}$ is optimal if it exists, and the problem is infeasible if no incumbent exists
- Else,
  - remove a node from $A$
  - label this node $t$
  - categorize $t$ into one of the four cases
- **Case 1:** Termination by infeasibility $\text{LP}(t)$ is infeasible. Terminate node $t$.
- **Case 2:** Termination by bound $\tilde{v}(t) \leq \hat{v}$. Terminate node $t$.
- **Case 3:** Termination by solution $\tilde{v}(t) > \hat{v}$ and $\tilde{x}(t)$ is integer. Terminate node $t$, set $\mathbf{x} \leftarrow \tilde{x}(t)$ and $\hat{v} \leftarrow \tilde{v}(t)$
- **Case 4:** Branching $\tilde{v}(t) > \hat{v}$ and $\tilde{x}(t)$ is not integer. Choose a variable that is not fixed in partial solution $x(t)$ and branch on all its possible values
- Increment solution counter $t \leftarrow t + 1$, goto Step 1
- Some areas of vagueness:
  - Which active node to choose in Step 1?
    * In principle, can select any active node
    * One potential rule: **depth first search** - select active node with the most components fixed (deepest in tree)
  - Which variable to branch on?
    * In principle, can select any variable not fixed at node’s partial solution
    * One potential rule: choose variable whose LP optimal value at that node is fractional and closest to integer
Branch and bound, illustrated

• LP relaxation of candidate problem at root node:
  \[
  \text{LP}^{(0)}: \quad \text{max} \quad 16x_1 + 22x_2 + 12x_3 + 8x_4 + 11x_5 + 19x_6 \\
  \text{s.t.} \quad 5x_1 + 7x_2 + 4x_3 + 3x_4 + 4x_5 + 6x_6 \leq 14 \\
  \quad 0 \leq x_i \leq 1 \quad i = 1, \ldots, 6
  \]
  Optimal solution: \( \tilde{v}^{(0)} = 44.4, \tilde{x}^{(0)} = (1, 0.43, 0, 0, 0, 1) \)
  \( \Rightarrow \) Case 4: branch on \( x_2 \)

• LP relaxation of candidate problem at root node \( \text{LP}^{(1)}: \)
  \[
  \text{max} \quad 16x_1 + 22x_2 + 12x_3 + 8x_4 + 11x_5 + 19x_6 \\
  \text{s.t.} \quad 5x_1 + 7x_2 + 4x_3 + 3x_4 + 4x_5 + 6x_6 \leq 14 \\
  \quad x_2 = 0 \\
  \quad 0 \leq x_i \leq 1 \quad i = 1, \ldots, 6
  \]
  Optimal solution: \( \tilde{v}^{(1)} = 44, \tilde{x}^{(1)} = (1, 0, 0.75, 0, 0, 1) \)
  \( \Rightarrow \) Case 4: branch on \( x_3 \)

• Solve \( \text{LP}^{(2)}: \)
  \( \tilde{v}^{(2)} = 43.25, \tilde{x}^{(2)} = (1, 0, 0, 0, 0.75, 1) \)
Case 4: branch on $x_5$

- Solve LP(3): $\tilde{v}^{(3)} = 43$, $\tilde{x}^{(3)} = (1, 0, 0, 1, 0, 1)$
- Solving LP(3) yields integer solution that is better than incumbent

⇒ Case 3: replace incumbent with $\tilde{x}^{(3)}$, terminate node 3

- Solve LP(4): $\tilde{v}^{(4)} = 42.8$, $\tilde{x}^{(4)} = (1, 0, 0, 1, 0, 1.83)$

⇒ Case 2: terminate node 4 by bound

- Solve LP(5): $\tilde{v}^{(5)} = 43.8$, $\tilde{x}^{(5)} = (1, 0, 1, 0, 0, 0.83)$
⇒ Case 4: branch on $x_6$

- Solve LP(6): $\bar{v}(6) = 41.6$, $\bar{x}(6) = (1, 0, 1, 0.33, 1, 0)$

⇒ Case 2: terminate node 6 by bound

- Solve LP(7): $\bar{v}(7) = 43.8$, $\bar{x}(7) = (0.8, 0, 1, 0, 1)$

⇒ Case 4: branch on $x_1$

- Solve LP(8): $\bar{v}(8) = 42$, $\bar{x}(8) = (0, 0, 1, 0, 1, 1)$
Case 2: terminate node 8 by bound

Case 2: terminate node 9 by infeasibility

Case 4: branch on $x_6$

And we keep on going in a similar manner until there are no active nodes left.
• We solved 29 LPs to get an optimal solution to the knapsack problem

• We found the optimal solution at the third iteration, but could not conclude that this solution was optimal until the 28th iteration

• What can we say about the quality of the solution we obtained at the third iteration?

Branch and bound family tree terminology

• Easiest to explain by a picture:

• Node 1 is the parent of nodes 3 and 4

• Nodes 1 and 2 are the children of node 0

Parent bounds

• Suppose we have a maximization integer linear program

• Example:
- $x^{(j)}$ = partial solution at node $j$
- $IP^{(j)}$ = node $j$’s candidate problem
- $LP^{(j)}$ = LP relaxation of node $j$’s candidate problem
- $v^{(j)}$ = optimal value of $IP^{(j)}$
- $\bar{v}^{(j)}$ = optimal value of $LP^{(j)}$

- $v^{(3)}$ = value of best completion of $x^{(3)}$
- $LP^{(3)} = LP^{(1)} +$ one additional variable fixed $\Rightarrow \bar{v}^{(3)} \leq \bar{v}^{(1)}$

$\Rightarrow v^{(3)} \leq \bar{v}^{(3)} \leq \bar{v}^{(1)}$

- $LP^{(1)}$ also provides an upper bound on the value of the best completion of $x^{(3)}$

**Parent bounds**

- For maximization ILPs, the optimal value of the LP relaxation of a parent node’s candidate problem provides an upper bound on the objective value of any completion of its children

- Similar reasoning for minimization ILPs

**Terminating nodes with parent bounds**

- Can use parent bounds to terminate some nodes even faster

- Example:

  $\hat{v}^{(0)} = 80$
  $\hat{v}^{(1)} = 75$
  $\hat{v}^{(2)} = 65$
  $\hat{v}^{(3)} = 80$

- $\alpha$, $\beta$, and $\gamma$ are active nodes
- Suppose new incumbent found at node 3 has value $\hat{v} = 70$
- Parent bound: all completions of node 2 have value $\leq 65$

$\Rightarrow$ No point in exploring $\beta$, $\gamma$, can terminate them immediately
Whenever branch and bound discovers a new incumbent solution, any active node whose parent bound is no better than the value of the new incumbent solution can be immediately terminated.

How good is the current incumbent?

- Sometimes just finding a feasible solution is difficult
- Would be nice to approximate how close a given solution is to optimal
- LP relaxations and parent bounds can help us do this

At node 3, we get a new incumbent with value $\hat{v} = 43$

Any solution that might improve upon the incumbent is a completion of some active partial solution.

⇒ Using parent bounds on $\alpha$, $\beta$, $\gamma$, we can conclude at this point in branch and bound that the optimal value must be at most

$$\max\{44.4, 44, 43.25\} = 44.4$$

What if we use the current incumbent as an approximation to the optimal solution?

The current incumbent is at most

$$\frac{(\text{best possible}) - (\text{best known})}{\text{best known}} = \frac{44.4 - 43}{43} = 3.25\%$$

below optimal

For maximization ILPs, we can obtain an upper bound on the optimal value by

- looking at the parent bound of all active nodes, and
- taking the highest parent bound

Can use this to obtain a bound in the error in using the incumbent as an approximation.
**Selecting active nodes**

- We used the depth first search rule in our illustration

- Other ideas:
  - **Best first** search selects at each iteration an active node with the best parent bound
  - **Depth forward best back** search selects
    * a deepest active node after a branching
    * an active node with best parent bound after a termination

**Branch and cut**

\[
\begin{align*}
\text{[P]} \quad \text{max} & \quad 3x_1 + 4x_2 \\
\text{s.t.} & \quad 5x_1 + 8x_2 \leq 24 \\
& \quad x_1, x_2 \geq 0 \\
& \quad x_1, x_2 \text{ integer}
\end{align*}
\]

\[
\begin{align*}
\text{[P']} \quad \text{max} & \quad 3x_1 + 4x_2 \\
\text{s.t.} & \quad 5x_1 + 8x_2 \leq 24 \\
& \quad x_1 + x_2 \leq 4 \\
& \quad x_1, x_2 \geq 0
\end{align*}
\]

- Add constraint \( x_1 + x_2 \leq 4 \)
- Note: this constraint holds for all integer feasible solutions, but cuts off feasible solutions from the LP relaxation [P']
- The constraint \( x_1 + x_2 \leq 4 \) is a valid inequality for [P]
- **Branch and cut** algorithms modify branch and bound by attempting to strengthen the LP relaxations of the candidate problems by adding valid inequalities
- Important: valid inequalities must hold for all feasible solutions to the full model, not just the candidate problems
• Added valid inequalities should cut off (render infeasible) the optimal solution to the LP relaxations of the candidate problems

• Sophisticated modern ILP codes are typically some variant of branch and cut

**Branch and bound**

• It is the starting point for all solution techniques for integer programming.
• Lots of research has been carried out over the past 40 years to make it more and more efficient.
• But, it is an art form to make it efficient. (We did get a sense why.)
• Integer programming is intrinsically difficult.
• How to do branching for general integer programs?