Lecture 2: Branch-and-Bound Method

(3 units)

Outline

- Linear program
- Simplex method
- Dual of linear programming
- Branch-and-bound method
- Software for integer programming

Linear Program

- Recall that the standard form of LP:

$$\begin{aligned} \min\, & c^T x \\ \text{s.t. } & Ax = b \\ & x \geq 0 \end{aligned}$$

  where $c \in \mathbb{R}^n$, A is an $m \times n$ matrix with full row rank, $b \in \mathbb{R}^m$.
- A polyhedron is a set of the form $\{x \in \mathbb{R}^n \mid Bx \geq d\}$ for some matrix B.
- Let $P \in \mathbb{R}^n$ be a given polyhedron. A vector $x \in P$ is an extreme point of P if there does not exist $y, z \in P$, and $\lambda \in (0, 1)$ such that $x = \lambda y + (1 - \lambda z)$.

Basic Solutions and Extreme Points

- Let $S = \{x \in \mathbb{R}^n \mid Ax = b,\ x \geq 0\}$, the feasible set of LP. Since A is full row rank, if the feaisble set is not empty, then we must $m \leq n$. WLOG, we assume that $m < n$.
- Let $A = (B, N)$, where B is an $m \times m$ matrix with full rank, i.e., $det(B) \neq 0$. Then, B is called a *basis*.
- Let $x = \begin{pmatrix} x_B \\ x_N \end{pmatrix}$.We have $Bx_B + Nx_N = b$. Setting $x_N = 0$, we have $x_B = B^{-1}b$. $x = \begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix}$ is called a *basic solution*. x_B is called *basic variables*, x_N is called *nonbasic variables*.
- If the basic solution is also feasible, this is, $B^{-1}b \geq 0$, then $x = \begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix}$ is called a *basic feasible solution*.

Basic Solutions and Extreme Points

- $\hat{x} \in S$ is an extreme point of S *if and only if* $\hat{x}$ is a basic feasible solution.
- Two extreme points are adjacent if they differ in only one basic variable.
- (Basic Theorem of LP) Consider the linear program $\min\{c^T x \mid Ax = b,\ x \geq 0\}$. If S has at least one extreme point and there exists an optimal solution, then there exists an optimal solution that is an extreme point.
- Proof (representation of polyhedron)
- The feasible set of standard form linear program has at least one extreme point.
- Therefore, we claim that the optimal value of a linear program is either $-\infty$, or is attained an extreme point (basic feasible solution) of the feasible set.

A naive algorithm for linear programming

- Let $\min\{c^T x \mid Ax = b,\ x \geq 0\}$ be a bounded LP
- Enumerate all bases $\mathcal{B} \in \{1, \ldots, , n\}$, $\begin{pmatrix} m \\ n \end{pmatrix} = O(n^m)$ many
- Compute associated basic solution $x = \begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix}$
- Return the one which has largest objective function value among the feasible basic solutions
- Running time is $O(n^m \cdot m^3)$!!!
- Are there more efficient algorithms?

Simplex method

- Simplex method was invented by George Dantzig (1914 – 2005) (father of linear programming)
- Suppose we have a basic feasible solution $\hat{x} = \begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix}$, $A = (B, N)$.
- Let $x \in S$ be any feasible solution of the LP. Let $x = \begin{pmatrix} x_B \\ x_N \end{pmatrix}$, $c = \begin{pmatrix} c_B \\ c_N \end{pmatrix}$. Then $Bx_B + Nx_N = b$ and so $x_B = B^{-1}b - B^{-1}Nx_N$

$$\begin{aligned} c^T x &= c_B^T x_B + c_N^T x_N \\ &= c_B^T B^{-1} b - c_B^T B^{-1} N x_N + c_N^T x_N \\ &= c^T \hat{x}^T + (c_N^T - c_B^T B^{-1} N) x_N \end{aligned}$$

- Let $r_N = c_N^T - c_B^T B^{-1} N$ (called *reduced cost*). If $r_N \geq 0$, then $c^T x \geq c^T \hat{x}^T$ and the current extreme point $\hat{x}$ is optimal

Simplex method

- Otherwise, there must exist an $r_i < 0$, we can let current nonbasic variable x_i become a basic variable $x_i > 0$ (entering variable).
- Suitably choosing basic variable to become a nonbasic variable (leaving variable), we can get a new basic feasible solution whose objective value is less than that of the current basic feasible solution $\hat{x}$.
- Geometrically, the above method (simplex method) moves from one extreme point to one of its adjacent extreme point.
- Since there are only a finite number of extreme points, the method terminates finitely at an optimal solution or detects that the problem is infeasible or it is unbounded.

Geometry of Simplex Method

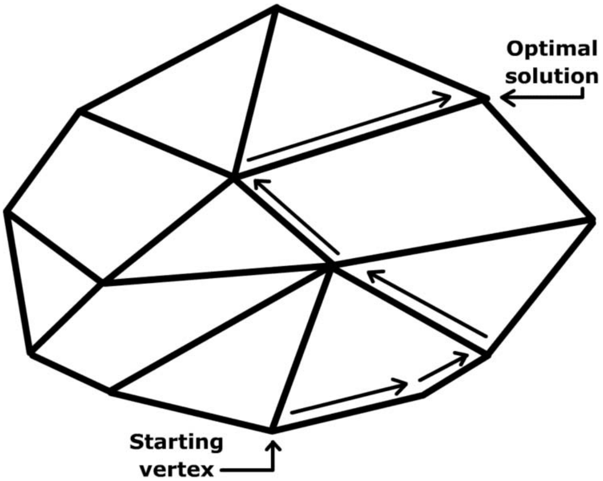


Figure: Simplex method

Simplex method

- Step 0: Compute an initial basis B and the basic feasible solution: $x = \begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix}$.
- Step 1: If $r_N = c_N^T - c_B^T B^{-1} N \geq 0$, stop, x is an optimal solution, otherwise go to Step 2.
- Step 2: Choose j satisfying $c_j^T - c_B^T B^{-1} a_j < 0$, if $\bar{a}_j = B^{-1} a_j \leq 0$, stop, the LP is infinite; otherwise, go to Step 3.
- Step 3. Compute the stepsize:

$$\lambda = \min\{\frac{\bar{b}_i}{\bar{a}_{ij}} \mid \bar{a}_{ij} > 0\} = \frac{\bar{b}_r}{\bar{a}_{rj}} \geq 0.$$

Let $x := x + \lambda d_j$, where $d_j = \begin{pmatrix} -B^{-1}a_j \\ e_j \end{pmatrix}$. Go to Step 1.

Simplex Tableau

$$
\begin{array}{cc|c}
x_B & x_N & \text{rhs} \\
\hline
B & N & b \\
\hline
c_B^T & c_N^T & 0 \\
\hline
\end{array}
\Longrightarrow
\begin{array}{cc|c}
x_B & x_N & \text{rhs} \\
\hline
I & B^{-1}N & B^{-1}b \\
\hline
0 & c_N^T - c_B^T B^{-1} N & -c_B^T B^{-1} b \\
\hline
\end{array}
$$

Simplex method: An example

- Consider the following LP:

$$\begin{aligned} \min\ & -7x_1 - 2x_2 \\ \text{s.t.}\ & -x_1 + 2x_2 + x_3 = 4, \\ & 5x_1 + x_2 + x_4 = 20, \\ & 2x_1 + 2x_2 - x_5 = 7, \\ & x \geq 0. \end{aligned}$$

- The initial tableau is

x_1	x_2	x_3	x_4	x_5	rhs
-1	2	1	0	0	4
5	1	0	1	0	20
2	2	0	0	-1	7
-7	-2	0	0	0	0

- Choose the initial basis: $B = (a_1, a_3, a_4)$, basic variable: $x_B = (x_1, x_3, x_4)^T$. The simplex tableau is:

x_1	x_2	x_3	x_4	x_5	rhs
0	3	1	0	$-\frac{1}{2}$	$7\frac{1}{2}$
0	-4	0	1	$2\frac{1}{2}$	$2\frac{1}{2}$
1	1	0	0	$-\frac{1}{2}$	$3\frac{1}{2}$
0	5	0	0	$-\frac{7}{2}$	$24\frac{1}{2}$

- The basic feasible solution is: $x_B = (x_1, x_3, x_4)^T = (3\frac{1}{2}, 7\frac{1}{2}, 2\frac{1}{2})^T$. Since $-\frac{7}{2} < 0$, choose x_5 as the entering variable, $\lambda = \frac{2\frac{1}{2}}{2\frac{1}{2}} = 1$, so x_4 is leaving variable, the new basic variable is $x_B = (x_1, x_3, x_5)$. The new tableau is

x_1	x_2	x_3	x_4	x_5	rhs
0	$\frac{11}{5}$	1	$\frac{1}{5}$	0	8
0	$-\frac{8}{5}$	0	$\frac{2}{5}$	1	1
1	$\frac{1}{5}$	0	$\frac{1}{5}$	0	4
0	$-\frac{5}{3}$	0	$\frac{7}{5}$	0	28

- Choose x_2 as the entering variable, compute $\lambda = \min\{\frac{8}{11/5}, \frac{4}{1/5}\} = \frac{40}{11}$. x_3 is the leaving variable, the new tableau is:

x_1	x_2	x_3	x_4	x_5	rhs
0	1	$\frac{5}{11}$	$\frac{1}{11}$	0	$\frac{40}{11}$
0	0	$\frac{8}{11}$	$\frac{6}{11}$	1	$\frac{75}{11}$
1	0	$-\frac{1}{11}$	$\frac{2}{11}$	0	$\frac{36}{11}$
0	0	$\frac{3}{11}$	$\frac{16}{11}$	0	$30\frac{2}{11}$

  Since $r_N = (\frac{3}{11}, \frac{16}{11}) \geq 0$, the current basic feasible solution $x = (\frac{36}{11}, \frac{40}{11}, 0, 0, \frac{75}{11})^T$ is the optimal solution with optimal value: $-30\frac{2}{11}$.

Dual theory: motivation

- Consider the following minimization problem with a single equality constraint:

$$(P) \qquad \begin{aligned} &\min f(x) \\ &\text{s.t. } \; h(x) = 0, \\ &\qquad\;\; x \in X, \end{aligned}$$

  where $h(x) = 0$ is a hard constraint and $x \in X$ is a "easy" constraint.
- How could we solve this problem? The idea is to relax the hard constraint $h(x) = 0$ and solve an easier problem.
- This can be done by incorporating the constraint into the objective function. A price is associated if the constraint is violated.

- Given a Lagrangian multiplier λ (price), the Lagrangian dual function defined by

$$d(\lambda) = \min_{x \in X} L(x, \lambda) := f(x) + \lambda h(x)$$

gives a lower bound for (P):

$$d(\lambda) \leq f(x), \quad \forall \text{ feasible solution of(P).}$$

- The best lower bound is found by solving

$$(D) \qquad \max_{\lambda} d(\lambda).$$

- (D) is called the Lagrangian dual problem of (P).

Dual of Linear Program

- Consider the standard LP:

$$(LP) \qquad \begin{aligned} &\min c^T x \\ &\text{s.t. } \; Ax = b, \; x \geq 0, \end{aligned}$$

- Associate multipliers $\lambda \in \mathbb{R}^m$ to $Ax = b$, the dual function is

$$\begin{aligned} d(\lambda) &= \min_{x \geq 0} \{c^T x + \lambda^T (b - Ax)\} = b^T \lambda + \min_{x \geq 0} (c - A^T \lambda)^T x \\ &= \begin{cases} b^T \lambda, & \text{if } A^T \lambda \leq c \\ -\infty, & \text{otherwise} \end{cases} \end{aligned}$$

- So, the dual problem is

$$(LD) \qquad \begin{aligned} &\max \; b^T \lambda \\ &\text{s.t. } \; A^T \lambda \leq c. \end{aligned}$$

Inequality Constraints

- Consider the standard LP:

$$(LP) \qquad \min c^T x \quad \text{s.t. } Ax \geq b,\ x \geq 0,$$

- Adding slack variables $s \geq 0$, we get

$$(A, -I) \begin{pmatrix} x \\ s \end{pmatrix} = b.$$

- This leads to dual constraints

$$(A, -I)^T \lambda \leq \begin{pmatrix} c \\ 0 \end{pmatrix}.$$

- Hence, the dual of LP is:

$$(LD) \qquad \max\ b^T \lambda \quad \text{s.t. } A^T \lambda \leq c,\ \lambda \geq 0.$$

Primal and Dual Forms

We can dualize general LPs as follows:

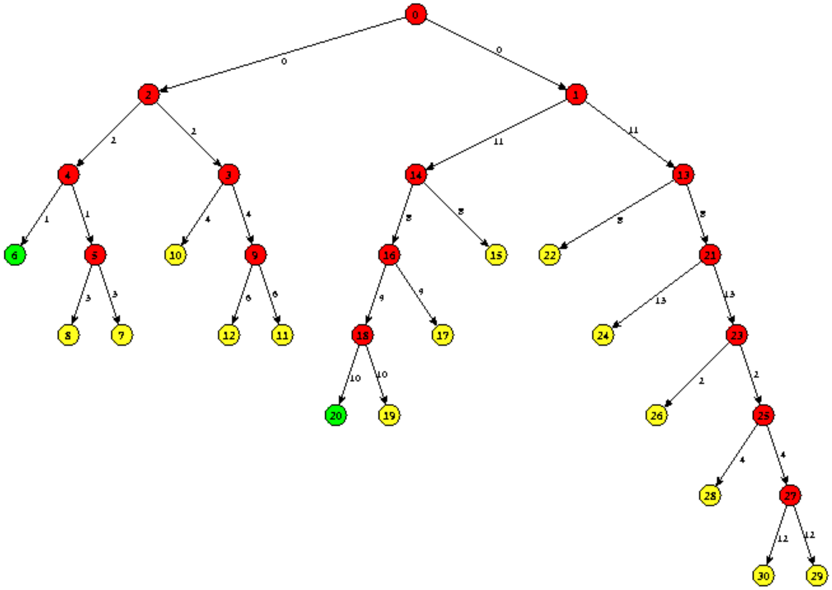
$$(LP)\left|\begin{array}{l} \min\ c^T x \\ \text{s.t. } Ax \begin{array}{c}\geq\\ \leq\\ =\end{array} b,\ x \begin{array}{c}\geq\\ \leq\\ \text{free}\end{array} 0 \end{array}\right. \Leftrightarrow (LD)\left|\begin{array}{l} \max\ b^T y \\ \text{s.t. } A^T y \begin{array}{c}\leq\\ \geq\\ =\end{array} c,\ \ y \begin{array}{c}\geq\\ \leq\\ \text{free}\end{array} 0 \end{array}\right.$$

Primal (Minimize)				Dual (Maximize)
Constraints	$\geq b_i$	$\Leftrightarrow$	≥ 0	Variables
	$\leq b_i$	$\Leftrightarrow$	≤ 0	
	$= b_i$	$\Leftrightarrow$	free	
Variables	≥ 0	$\Leftrightarrow$	$\leq c_j$	Constraints
	≤ 0	$\Leftrightarrow$	$\geq c_j$	
	free	$\Leftrightarrow$	$= c_j$	

Branch-and-Bound Method

- Branch-and-bound strategy:
  - Solve the linear relaxation of the problem. If the solution is integer, then we are done. Otherwise create two new subproblems by branching on a fractional variable.
  - A node (subproblem) is not active when any of the following occurs:
    (1)The node is being branched on;
    (2) The solution is integral;
    (3) The subproblem is infeasible;
    (4) You can fathom the subproblem by a bounding argument.
  - Choose an active node and branch on a fractional variable. Repeat until there are no active subproblems.

Branch-and-bound search tree:

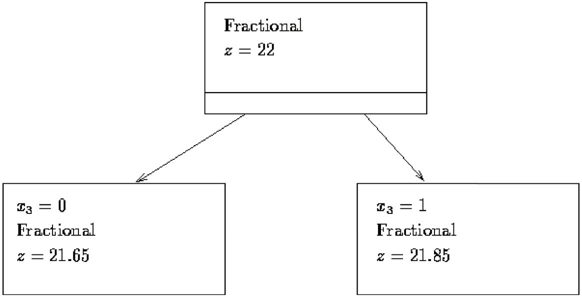


- Example. Consider the following 0-1 knapsack problem:

$$\begin{aligned} \max\ & 8x_1 + 11x_2 + 6x_3 + 4x_4 \\ \text{s.t.}\ & 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14 \\ & x \in \{0,1\}^4. \end{aligned}$$

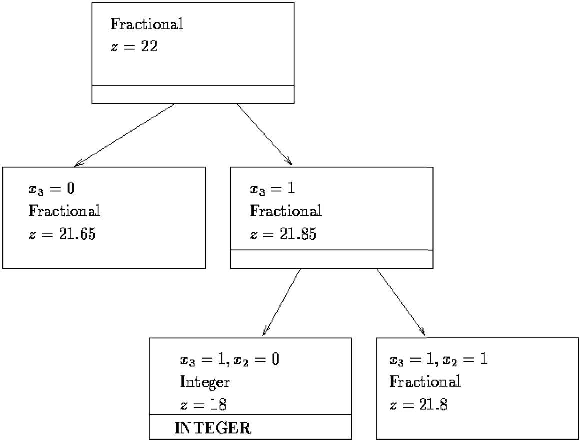
- The linear relaxation solution is $x = (1, 1, 0.5, 0)$ with a value of 22. The solution is not integral.
- Choose x_3 to branch. The next two subproblems will have $x_3 = 0$ and $x_3 = 1$ respectively.

- The search tree is:



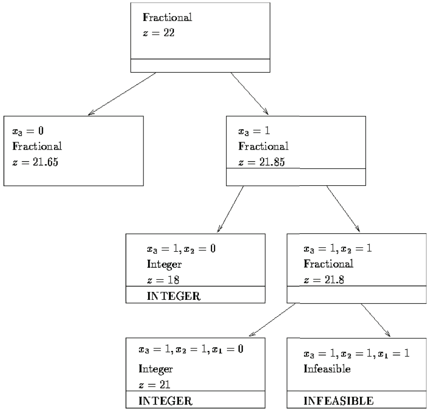
- The linear relaxation solutions to the two subproblems are
  - $x_3 = 0$: $x = (1, 1, 0, 0.667)$, objective value=21.65;
  - $x_3 = 1$: $x = (1, 0.714, 1, 0)$, objective value=21.85.
- At this point we know that the optimal value is no more than 21.85 (we actually know it is less than or equal to 21. but we still do not have any feasible integer solution. So, we will take a subproblem and branch on one of its variables.

- Choose the node with $x_3 = 1$ and branch on x_2. The branch tree is:



- The linear relaxation solutions are
  - $x_3 = 1$, $x_2 = 0$: $x = (1, 1, 1, 1)$, objective value=18;
  - $x_3 = 1$: $x_2 = 1$: $x = (0.6, 1, 1, 0)$, objective value=21.8.

- Choose node with $x_3 = 1$, $x_2 = 0$ and branch on x_1. The search tree is



- The linear relaxation solutions are
  - $x_3 = 1$, $x_2 = 0$, $x_1 = 0$: $x = (0, 1, 1, 1)$, objective value=21;
  - $x_3 = 1$: $x_2 = 1$: $x_1 = 1$: infeasible.
- The optimal solution is $x = (0, 1, 1, 1)$.

Solving 0-1 Knapsack by B&B

- 0-1 Knapsack Problem

$$(01KP) \qquad \max\{c^T x \mid a^T x \leq b, x \in \{0,1\}^n\}.$$

- To solve the LP relaxation of (01KP), you need only be greedy! Assume that the variables have been ordered such that

$$c_1/a_1 \geq c_2/a_2 \geq \cdots \geq c_n/a_n. \tag{1}$$

Let s be the maximum index k such that

$$\sum_{j=1}^{k} a_j \leq b. \tag{2}$$

- Theorem (Dantzig (1957)): *The optimal solution to the continuous relaxation of* $(01KP)$ *is*

$$\begin{aligned} w_j &= 1,\ j = 1, \dots, s, \\ w_j &= 0,\ j = s+2, \dots, N, \\ w_{s+1} &= (b - \sum_{j=1}^{s} a_j)/a_{s+1}. \end{aligned}$$

- If c_j, $j = 1, \dots, N$, are positive integers, then an upper bound of the optimal value of (LKP) is given by

$$UB = \sum_{j=1}^{s} c_j + \lfloor (b - \sum_{j=1}^{s} a_j) c_{s+1}/a_{s+1} \rfloor, \tag{3}$$

How to Branch?

- We want to divide the current problem into two or more subproblems that are easier than the original. A commonly used branching method:

$$x_i \leq \lfloor x_i^* \rfloor, \; x_i \geq \lceil x_i^* \rceil,$$

  where x_i^* is a fractional variable.
- Which variable to branch? A commonly used branching rule: Branch the most fractional variable.
- We would like to choose the branching that minimizes the sum of the solution times of all the created subproblems.
- How do we know how long it will take to solve each subproblem?

  Answer: We don't.

  Idea: Try to predict the difficulty of a subproblem.
- A good branching rule: The value of the linear programming relaxation changes a lot!

Which Node to Select?

- An important choice in branch and bound is the strategy for selecting the next subproblem to be processed.
- Goals: (1) Minimizing overall solution time. (2) Finding a good feasible solution quickly.
- Some commonly used search strategies:
  - Best First
  - Depth-First
  - Hybrid Strategies
  - Best Estimate

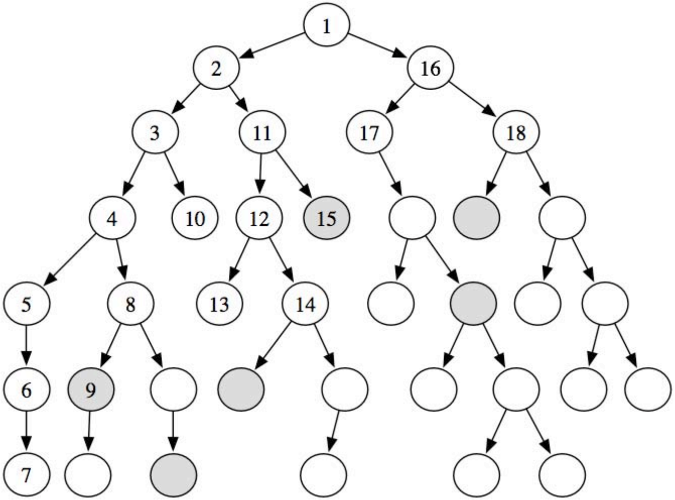
The Best First Approach

- One way to minimize overall solution time is to try to minimize the size of the search tree. We can achieve this by choosing the subproblem with the best bound (lowest lower bound if we are minimizing).
- Drawbacks of Best First
  - Doesn't necessarily find feasible solutions quickly since feasible solutions are "more likely" to be found deep in the tree
  - Node setup costs are high. The linear program being solved may change quite a bit from one node evaluation to the next
  - Memory usage is high. It can require a lot of memory to store the candidate list, since the tree can grow "broad"

The Depth First Approach

- The depth first approach is to always choose the deepest node to process next. Just dive until you prune, then back up and go the other way
- This avoids most of the problems with best first: The number of candidate nodes is minimized (saving memory). The node set-up costs are minimized
- LPs change very little from one iteration to the next. Feasible solutions are usually found quickly
- Drawback: If the initial lower bound is not very good, then we may end up processing lots of non-critical nodes.
- Hybrid Strategies: Go depth-first until you find a feasible solution, then do best-first search

The nodes expanded in depth-first branch-and-bound search:



Software for integer programming

- Matlab code: `bintprog` for solve binary integer linear programming problems.
- Commercial optimization software `CPLEX` and `Gurobi` for solving mixed integer linear and quadratic programming problems.
- Two simple ways of calling CPLEX MIP solver:
  - Optimization Programming Language (OPL) in CPLEX Optimization Studio.
  - CPLEX Matlab interface.
- Examples using Matlab and CPLEX