NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

AI6101 Introduction to AI and AI Ethics

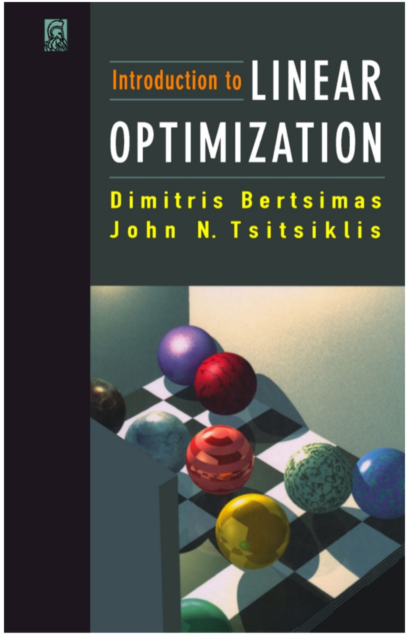
Linear Programming

Assoc Prof Bo AN

www.ntu.edu.sg/home/boan
Email: boan@ntu.edu.sg
Office: N4-02b-55

Lesson Outline

- Introduction
- Linear algebra recap
- Linear programming
- Simplex method
- Integer programming



Introduction

What is Linear Programming?

- A central topic in optimization
- A powerful and general problem solving method
  - shortest path, max flow, min cost flow, generalized flow, multicommodity flow, MST, matching, 2-person zero sum games

Why Linear Programming?

- Applicability: There are many real-world applications that can be modeled as linear programming;
- Solvability: There are theoretically and practically efficient techniques for solving large-scale problems.

Introduction-Applications

Linear Programming Applications:

- Computer science. Compiler register allocation, data mining.
- Electrical engineering. VLSI design, optimal clocking.
- Economics. Equilibrium theory, two-person zero-sum games.
- Environment. Water quality management.
- Finance. Portfolio optimization.
- Marketing. Direct mail advertising.
- Manufacturing. Production line balancing, cutting stock.
- Operations research. Airline crew assignment, vehicle routing.
- Telecommunication. Network design, Internet routing.

Manufacturing Example

A large factory makes tables and chairs.
Each table returns a profit of $200 and takes 1 unit of metal and 3 units of wood.
Each chair returns a profit of $100 and takes 2 units of metal and1 unit of wood.

The factory has 6K units of metal and 9K units of wood.

How many tables and chairs should the factory make to maximize profit?

If all table: 3k tables = $600k; If all chair: 3k chairs = $300k;
If 2.4k tables+1.8k chairs = $660k

Linear Algebra Recap

Vector

- We use the notation $x \in \mathrm{R}^n$ to denote a vector with n entries, x_i denotes the ith element of x
- By convention, $x \in \mathrm{R}^n$ represents a column vector; to indicate a row vector, we use x^T

Matrices

- We use the notation $A \in \mathrm{R}^{\mathrm{m} \times n}$ to denote a matrix with m rows and n column

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, x^T = \begin{bmatrix} x_1 & x_2 & \dots & x_3 \end{bmatrix}, A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

Linear Algebra Recap

Matrix addition/subtraction

- defined as addition or subtraction of the elements
- for $A, B \in \mathrm{R}^{m \times n}$,

$$C \in \mathrm{R}^{m \times n} = A + B \Leftrightarrow c_{ij} = a_{ij} + b_{ij}$$

Matrix transpose

- switch rows and columns $D \in \mathrm{R}^{n \times m} = A^T \Leftrightarrow d_{ij} = a_{ji}$

Matrix multiplication

- for $A \in \mathrm{R}^{m \times n}, B \in \mathrm{R}^{n \times p}$,

$$C \in \mathrm{R}^{m \times p} = AB \Leftrightarrow c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

Linear Algebra Recap

Matrix multiplication properties

- Associative: $A(BC) = (AB)C$
- Distributive: $A(B + C) = AB + AC$
- Not commutative: $AB \neq BA$
- Transpose of product: $(AB)^T = B^T A^T$

Matrix inverse

- For a square matrix $A \in \mathrm{R}^{n \times n}$, inverse A^{-1} is the matrix such that

$$A^{-1}A = I = AA^{-1}$$

Vector norm

- For $x \in \mathrm{R}^n$, we use $||x||_2$ to denote the *Euclidean norm* of x

$$||x||_2 = \sqrt{x^T x}$$

Linear Programming

The linear model consists of the following components:

- maximizing (or minimizing) a linear objective function
- of n variables $x_1, \dots , x_n$
- subject to a set of constraints expressed by linear inequalities or equations

Example of previous case

$$
\begin{aligned}
\max \quad & 200x_1 + 100x_2 \\
\text{subject to} \quad & \\
& x_1 + 2x_2 \leq 6000 \\
& 3x_1 + x_2 \leq 9000 \\
& x_1 \geq 0 \\
& x_2 \geq 0
\end{aligned}
$$

Standard Form

$$\min \sum_{i=1}^{n} c_i x_i$$

objective function

subject to

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i, \qquad i = 1 \dots m$$

constraints

$$x_j \ge 0, \qquad j = 1 \dots n$$

non-negativity constraints

Standard Form (Matrices)

$$\min \quad \sum_{i=1}^{n} c_i x_i \text{ subject to } \begin{cases} Ax = b \\ x \geq 0 \end{cases}$$

where

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \qquad c = \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix}$$

$$b = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} \qquad \mathrm{A} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

Equivalent Forms

- A maximization problem can be expressed as a minimization problem.

$$\max c^T x \Leftrightarrow \min -c^T x$$

- An equality can be represented as a pair of inequalities.

$$a_i^T x = b_i \Leftrightarrow \begin{cases} a_i^T x \leq b_i \\ a_i^T x \geq b_i \end{cases}$$

- By adding a slack variable, an inequality can be represented as a combination of equality and non-negativity constraints.

$$a_i^T x \leq b_i \Leftrightarrow a_i^T x + s_i = b_i, s_i \geq 0$$

- Non-positivity constraints can be expressed as non-negativity constraints.
- x may be unrestricted in sign.

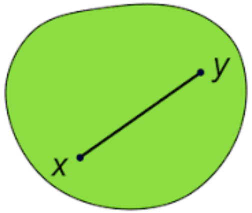
replace x_j by $x_j^+ - x_j^-$, adding the constraints $x_j^+, x_j^- \geq 0$

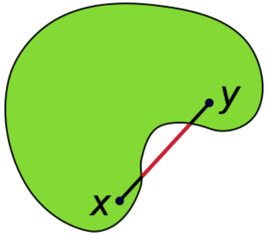
Definitions

$$\min \sum_{i=1}^{n} c_i x_i \text{ subject to } \begin{cases} Ax = b \\ x \geq 0 \end{cases}$$

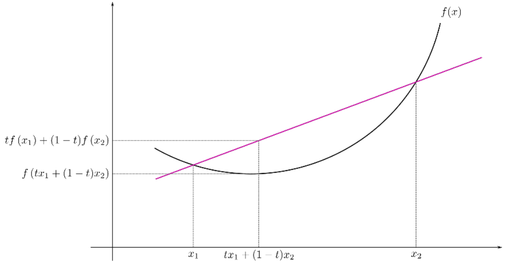
- Any vector x such that $Ax = b$ is called a solution
- A solution x satisfying $x \geq 0$ is called a feasible solution
- An LP with feasible solutions is called feasible; otherwise it is said to be infeasible.
- A feasible solution x^* is called optimal, if $c^T x^* \leq c^T x$ for all feasible solution x
- A feasible LP with no optimal solution is unbounded

Definitions





$$\min \sum_{i=1}^{n} c_i x_i \text{ subject to } \begin{cases} Ax = b \\ x \geq 0 \end{cases}$$



- Given points $x, y \in \mathrm{R}^n$, a point $z \in \mathrm{R}^n$ is a convex combination of x, y if $z = \lambda x + (1-\lambda)y$ for some $\lambda \in [0, 1]$
- A set $X \subseteq \mathrm{R}^{\mathrm{n}}$ is convex if the convex combination of any two points in X is also in X
- A function $f: \mathrm{R}^n \rightarrow \mathrm{R}^n$is convex if for all points $x, y \in \mathrm{R}^n$ and all $\lambda \in [0,1]$ we have $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$
- Fact: The intersection of two convex sets is convex

Definitions

$$\min \sum_{i=1}^{n} c_i x_i \text{ subject to } \begin{cases} Ax = b \\ x \geq 0 \end{cases}$$

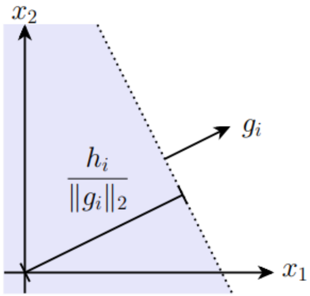
- A set $H \subseteq \mathrm{R}^{\mathrm{n}}$ is a hyperplane if $H = \{x \in \mathrm{R}^n | a^T x = b\}$ for some nonzero $a \in \mathrm{R}^n$ and some $b \in \mathrm{R}^n$
- A set $H' \subseteq \mathrm{R}^{\mathrm{n}}$ is a halfspace if $H' = \{x \in \mathrm{R}^n | a^T x \geq b\}$ for some nonzero $a \in \mathrm{R}^n$ and some $b \in \mathrm{R}^n$
- A polyhedron in R^n is the intersection of finitely many halfspaces
- A polytope is a bounded polyhedron, that is, a polyhedron P for which there exist $B \in \mathrm{R}^n$ such that $||x||_2 \leq B$ for all $x \in P$
- Both hyperplanes and halfspaces are convex sets
- Both polyhedron and polytopes are convex

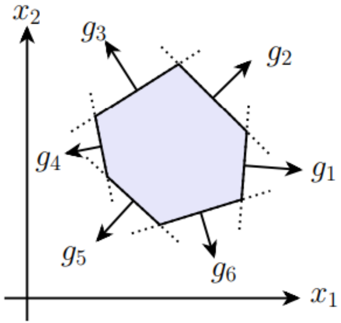
Geometry of LP

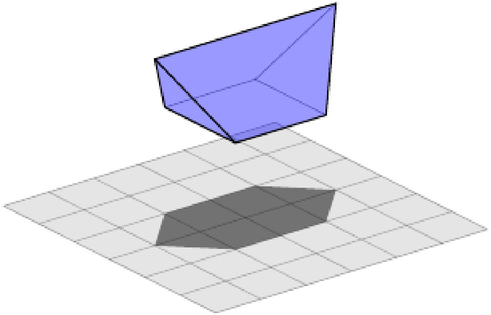
Consider the inequality constraints of the linear program written as:

$$\max \sum_{i=1}^{n} c_i x_i \text{ subject to } g_i^T x \leq h_i, i = 1, \dots, m$$

- Each constraint $g_i^T x \leq h_i$ represents a hyperplane or halfspace
- Multiple half-space constraints, $g_i^T x \leq h_i, i = 1, \dots, m$, define a convex polyhedron or polytope







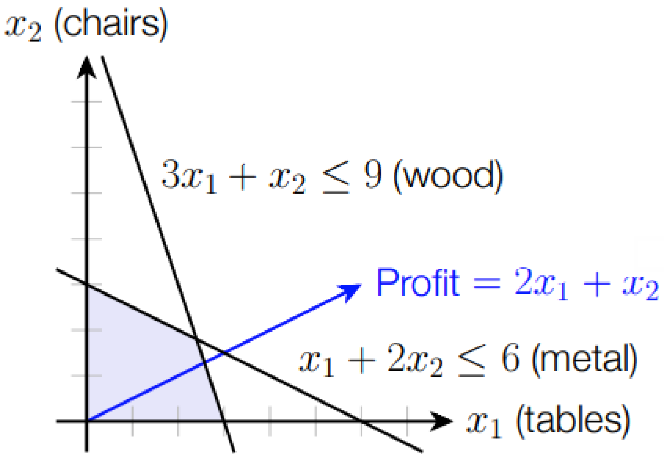
Solving Linear Programming

- Difficulty: Large number of solutions
  - Choose the best solution among 2n or n! possibilities: all solutions cannot be enumerated
  - Complexity of studied problems: often NP-complete
- Solving methods:
  - Optimal solutions:
    - Graphical method (2 variables only)
    - Simplex method
  - Approximations:
    - Theory of duality (assert the quality of a solution)
    - Approximation algorithms

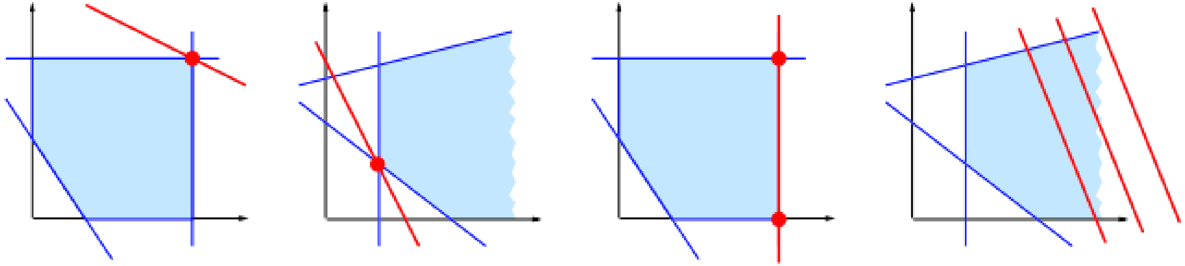
Graphical Method

- The constraints of an LP define a zone of solutions
- The best point of the zone corresponds to the optimal solution
- For problem with 2 variables, easy to draw the zone of solutions and to find the optimal solution graphically

$$\begin{aligned} & \max \quad 2x_1 + x_2 \\ \text{subject to} & \\ & x_1 + 2x_2 \leq 6 \\ & 3x_1 + x_2 \leq 9 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \end{aligned}$$



Optimal Solutions Cases:

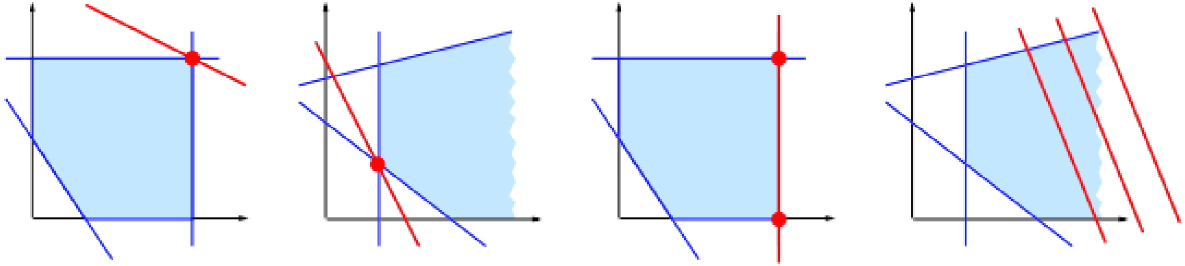


Three different possible cases:

- A single optimal solution
- Infinite optimal solutions
- No optimal solutions

Note that, if an optimal solutions exist, it always occur at a corner of the polytope.

Optimal Solutions Cases:



Note that:

- If two points a, b are feasible solutions, then so is $(a+b)/2$
- if an optimal solutions exist, it always occur at a corner of the polytope

We define:

- Extreme point: feasible solution x that can't be written as $(a+b)/2$ for any two distinct feasible solutions a and b

Optimal Solutions Cases:

Extreme point property

- If there exists an optimal solution to (P), then there exists one that is an extreme point solution
- Only need to consider finitely many possible solutions

Challenge

- Number of extreme points can be exponential
- Consider n-dimensional hypercube

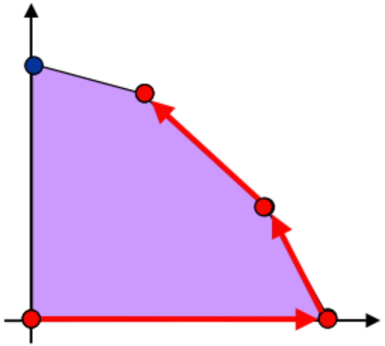
Greedy

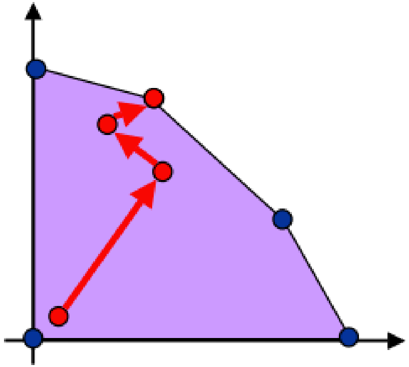
- Local optima are global optima
- Extreme point is optimal if no neighboring extreme point is better.

Solving Linear Programming

- Geometric method impossible in higher dimensions

- Algebraical methods:
  - Simplex method
  - Interior point methods
    - Ellipsoid Algorithm
    - Karmarkar’s Algorithm





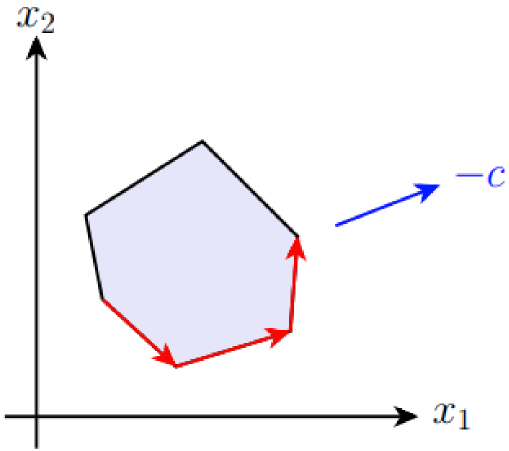
Simplex Algorithm

Simplex algorithm (George Dantzig, 1947)

- Developed shortly after World War II in response to logistical problems
- Used for 1948 Berlin airlift
- Most popular method to solve linear programs

Basic idea:

- Start at some extreme point
- Pivot from one extreme point to a neighboring one
- In directions of decreasing cost
- Repeat until optimal



Implementation: linear algebra

Simplex Algorithm

Example linear program:

$$\begin{aligned} x_1 &+ x_2 \leq 3 \\ -x_1 &+ 3x_2 \leq 1 \\ & x2 \leq 3 \\ x_1 &+ x_2 = z \end{aligned}$$

The last line is the objective function we are trying to maximize

We assume:

- all the constraints are $\leq$
- All the values of the variable must be ≥ 0

Simplex Algorithm

Rewrite into a system of equations by introducing non-negative slack variables: $x_3, x_4, x_5 \geq 0$

$$\begin{array}{rcrcl}
x_1 & + & x_2 & + x_3 & = 3 \\
-x_1 & + & 3x_2 & + x_4 & = 1 \\
 & & x2 & + x_5 & = 3 \\
x_1 & + & x_2 & & = z
\end{array}$$

We can easily find a solution:

$$x_3 = 3, x_4 = 1, x_5 = 3, x_1 = x_2 = 0$$

Our objective now is $z = 0$

Simplex Algorithm

Rewrite to put the non-zero values on the left-hand side

$$\begin{aligned} x_3 &= 3 - x_1 - x_2 \\ x_4 &= 1 + x_1 - 3x_2 \\ x_5 &= 3 - x_2 \\ z &= 0 + x_1 + x_2 \end{aligned}$$

This is called a tableau: Right-hand side variables are all 0, left hand side may be non-zero.

The left hand side variables are called basic variables.

Which variables are candidates for increasing to increase z?

- Those with positive coefficients in the objective

Simplex Algorithm

$$\begin{aligned} x_3 &= 3 \quad - x_1 \quad - x_2 \\ x_4 &= 1 \quad + x_1 \quad - 3x_2 \\ x_5 &= 3 \qquad\qquad\ - x_2 \\ z &= 0 \quad + x_1 \quad + x_2 \end{aligned}$$

We choose x_2 to increase as long as none of the basic variables become negative. That is

$$\begin{aligned} 3 - x_2 \geq 0 &\Rightarrow x_2 \leq 3 \\ 1 - 3x_2 \geq 0 &\Rightarrow x_2 \leq 1/3 \\ 3 - x_2 \geq 0 &\Rightarrow x_2 \leq 3 \end{aligned}$$

So we set $x_2 = 1/3$, call it an entering variable

Simplex Algorithm

If we set $x_2 = 1/3$, then $x_4 = 0$

We rewrite by move x_2 to left hand and x_4 to right hand

$$\begin{aligned}
x_3 &= 8/3 \quad - (4/3)x_1 \quad + (1/3)x_4 \\
x_2 &= 1/3 \quad + (1/3)x_1 \quad - (1/3)x_4 \\
x_5 &= 8/3 \quad - (1/3)x_1 + (1/3)x_4 \\
z &= 1/3 \quad + (4/3)x_1 \quad - (1/3)x_4
\end{aligned}$$

We choose x_1 to increase as long as none of the basic variables become negative. That is

$$\begin{aligned}
&8/3 \quad - (4/3)x_1 \geq 0 \Rightarrow x_1 \leq 2 \\
&1/3 \quad + (1/3)x_1 \geq 0 \Rightarrow x_1 \geq -1 \\
&8/3 \quad - (1/3)x_1 \geq 0 \Rightarrow x_1 \leq 8
\end{aligned}$$

So we set $x_1 = 2$

Simplex Algorithm

If we set $x_1 = 2$, then $x_3 = 0$

We rewrite by move x_2 to left hand and x_3 to right hand

$$\begin{aligned} x_1 &= 2 \quad - (3/4)x_3 \quad + (1/4)x_4 \\ x_2 &= 1 \quad - (1/4)x_3 \quad - (1/4)x_4 \\ x_5 &= 2 \quad + (1/4)x_3 \quad + (1/4)x_4 \\ z &= 3 \quad - x_3 \end{aligned}$$

Now we can find all the coefficients in the objective function are ≤ 0

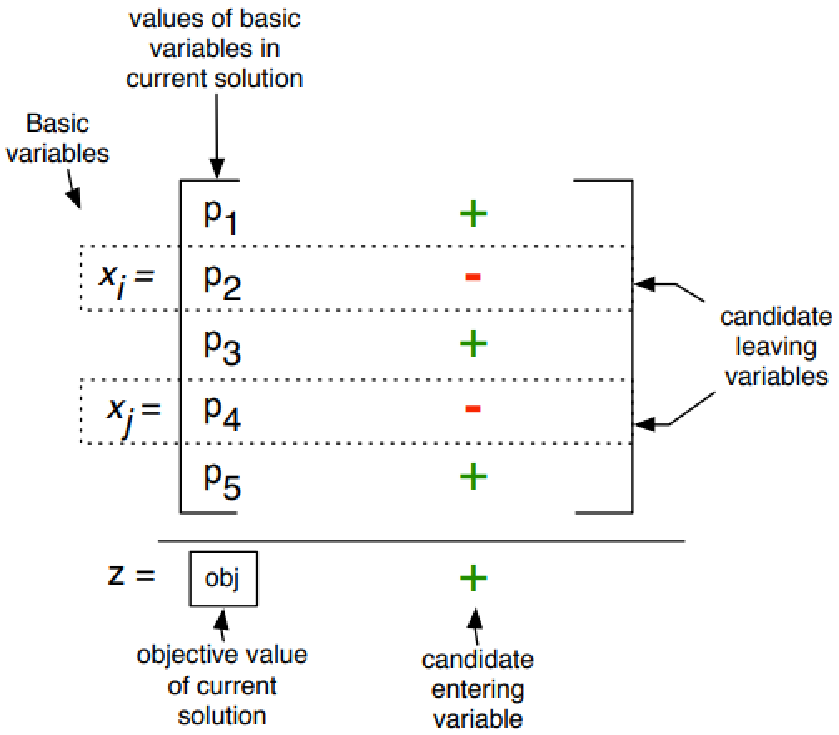
So the optimal value is $x_1 = 2, x_2 = 1$

Simplex Algorithm

General steps:

1. Write LP with slack variables (slack vars = initial solution)
2. Choose a variable v in the objective with a positive coefficient to increase
3. Among the equations in which v has a negative coefficient q_{iv} , choose the strictest one
   - This is the one that minimizes $-p_i/q_{iv}$ because the equations are all of the form $x_i = p_i + q_{iv}x_v$.
4. Re-write the strictest equation to put v on the left-hand side, and substitute for v everywhere else.
5. If all the coefficients are ≤ 0 in objective, we're done; otherwise, jump back to step 2.

Simplex Algorithm


values of basic variables in current solution
Basic variables
$x_i =$
$x_j =$
p_1
+
p_2
-
p_3
+
p_4
-
p_5
+
candidate leaving variables
$z =$
obj
+
objective value of current solution
candidate entering variable

Integer Linear Programming

Integer Linear Programming (ILP)

$$\min \sum_{i=1}^{n} c_i x_i \text{ subject to } \begin{cases} Ax = b \\ x \geq 0 \\ x \in \mathrm{Z}^n \end{cases}$$

Binary Integer Linear Programming (BILP)

$$\min \sum_{i=1}^{n} c_i x_i \text{ subject to } \begin{cases} Ax = b \\ x \geq 0 \\ x \in \{0,1\}^n \end{cases}$$

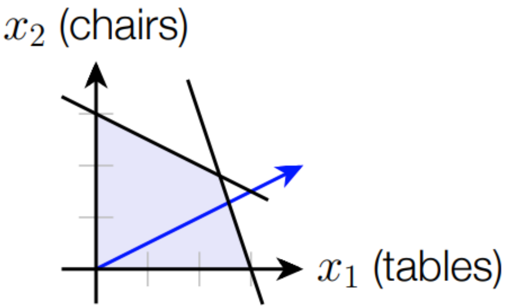
Mixed Integer Linear Programming (MILP)

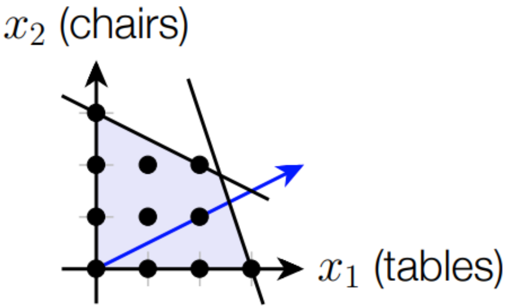
$$\min \sum_{i=1}^{n} c_i x_i + \sum_{j=1}^{m} h_j y_j \text{ subject to } \begin{cases} Ax + Gy = b \\ x \geq 0, y \geq 0 \\ y \in \mathrm{Z}^n \end{cases}$$

Recall LP

A large factory makes tables and chairs.
Each table returns a profit of $200 and takes 1 unit of metal and 3 units of wood
Each chair returns a profit of $100. Each chair takes 2 units of metal and1 unit of wood.

What if the factory has 6 units of metal and 9 units of wood.





Integer Linear Programming

Difficulty:

- NP-complete
- Non-convex: set of all integers is not a convex set

Applications:

- Path planning with obstacles
- Many problems in game theory
- Constraint satisfaction problems
- (Exact) most likely assignment in graphical models
- Scheduling and unit commitment
- Kidney exchange

Integer Linear Programming

We focus on binary integer programming for simplicity in this lecture

$$\min \sum_{i=1}^{n} c_i x_i \text{ subject to } \begin{cases} Ax = b \\ x \geq 0 \\ x \in \{0,1\}^n \end{cases}$$

This is just for ease of presentation, we will discuss how to adapt all these methods for general integer variables

Techniques we present are actually largely applicable to any mixed integer programming problem with convex objective and constraints (other than integer constraint)

Solving ILP

$$\min \sum_{i=1}^{n} c_i x_i \text{ subject to } \begin{cases} Ax = b \\ x \geq 0 \\ x \in \{0,1\}^n \end{cases}$$

The naïve solution: 2^n possible assignments of all n variables, just try each one, return solution with minimum objective value out of those that satisfy constraints

In the worst case, we can't do any better than this, but often it is possible to solve the problem much faster in practice

Solving ILP

We relax the constraint $x \in \{0,1\}^n$ to be $x \in [0,1]^n$:

$$\min \quad \sum_{i=1}^{n} c_i x_i \text{ subject to } \begin{cases} Ax = b \\ x \geq 0 \\ x \in [0,1]^n \end{cases}$$

Key point #1: if the solution to this linear program x^* has all integer values, then it is also the solution to the integer program

Key point #2: the optimal objective for the linear program will be lower than that of the binary integer program

Branch and Bound

LP relaxation is a quickly-computable approximation which gives us a lower bound on the true solution: sounds a lot like an admissible heuristic…

This leads us to the branch and bound algorithm: this is just greedy informed search (i.e., $f(s) = h(s)$, no path cost, just heuristic cost), applied using LP relaxation as the heuristic

Repeat:

1. Choose relaxed problem from frontier with lowest cost
2. If solution is not integer valued, pick a non-integer variable x_i and add problems with additional constraints $x_i = 0$ and $x_i = 1$
3. If solution is integer valued, return

Branch and Bound

Function: Solve-Relaxation(S):

- Solve linear program plus additional constraints in S
- Return (objective value f^* , solution x^*, and constraint set S)

Algorithm: Branch-and-Bound

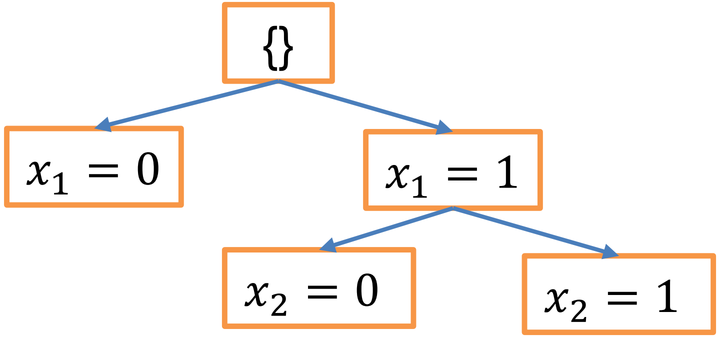
- Push Solve-Relaxation({}) on to frontier set
- Repeat while frontier is not empty:
    1. Get lowest cost solution from frontier: (f, x, S)
    2. If x is integer valued, return x
    3. Else, choose some x_i not integer valued and add Solve-Relaxation($S \cup \{x_i = 0\}$),Solve-Relaxation($S \cup \{x_i = 1\}$), to the frontier
    4. Eliminating subtrees if the lower bound of a subtree is no less than an integer valued solution

Branch and bound

$$\begin{array}{ll} \text{minimize} & 2x_1 \quad + \quad x_2 - 2x_3 \\ \text{subject to} & 0.7\mathrm{x}_1 + 0.5\mathrm{x}_2 + \mathrm{x}_3 \geq 1.8 \\ & x_i \in \{0,1\}, i = 1,2,3 \end{array}$$

Relax:

$$\begin{array}{ll} \text{minimize} & 2x_1 \quad + \quad x_2 - 2x_3 \\ \text{subject to} & 0.7\mathrm{x}_1 + 0.5\mathrm{x}_2 + \mathrm{x}_3 \geq 1.8 \\ & x_i \in [0,1], i = 1,2,3 \end{array}$$



$(f^* = -1/7, x^* = [3/7,1,1], S = \{\})$

$(f^* = 0.2, x^* = [1,0.2,1], S = \{x_1 = 1\})$

$(f^* = 1, x^* = [1,1,1], S = \{x_1 = 1, x_2 = 1\})$

$(f^* = \infty, x^* = \emptyset, S = \{x_1 = 0\})$

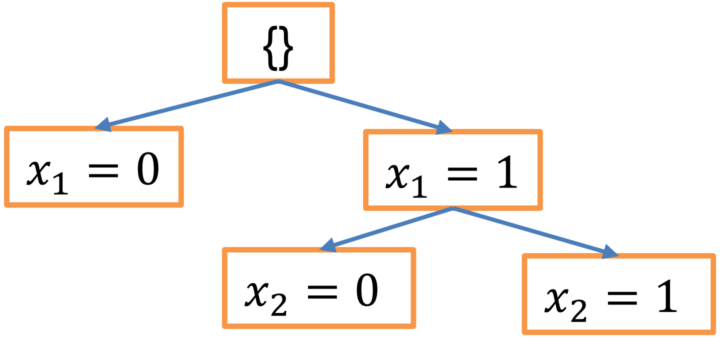
$(f^* = \infty, x^* = \emptyset, S = \{x_1 = 1, x_2 = 0\})$

Branch and Bound

$$\begin{aligned} &\text{minimize} && 2x_1 \quad + \quad x_2 - 2x_3 \\ &\text{subject to} && 0.7\mathrm{x}_1 + 0.5\mathrm{x}_2 + \mathrm{x}_3 \geq 1.8 \\ & && x_i \in \{0,1\}, i = 1,2,3 \end{aligned}$$

Relax:

$$\begin{aligned} &\text{minimize} && 2x_1 \quad + \quad x_2 - 2x_3 \\ &\text{subject to} && 0.7\mathrm{x}_1 + 0.5\mathrm{x}_2 + \mathrm{x}_3 \geq 1.8 \\ & && x_i \in [0,1], i = 1,2,3 \end{aligned}$$



$(f^* = -0.143, x^* = [0.43,1,1], S = \{\})$

$(f^* = 0.2, x^* = [1,0.2,1], S = \{x_1 = 1\})$

$(f^* = 1, x^* = [1,1,1], S = \{x_1 = 1, x_2 = 1\})$

$(f^* = \infty, x^* = \emptyset, S = \{x_1 = 0\})$

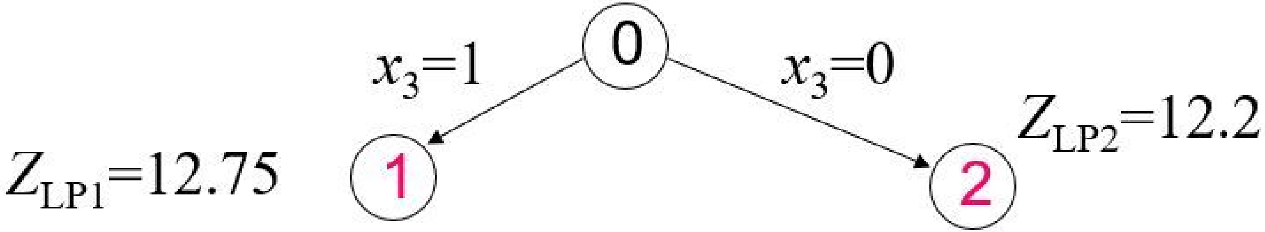
$(f^* = \infty, x^* = \emptyset, S = \{x_1 = 1, x_2 = 0\})$

Branch and Bound - Example

Maximize $4x_1+9x_2+6x_3$

Subject to $5x_1+8x_2+6x_3 \leq 12$

x_1, x_2, x_3 are binary variables



Maximize $4x_1+9x_2+6$

s.t. $5x_1+8x_2 \leq 6$

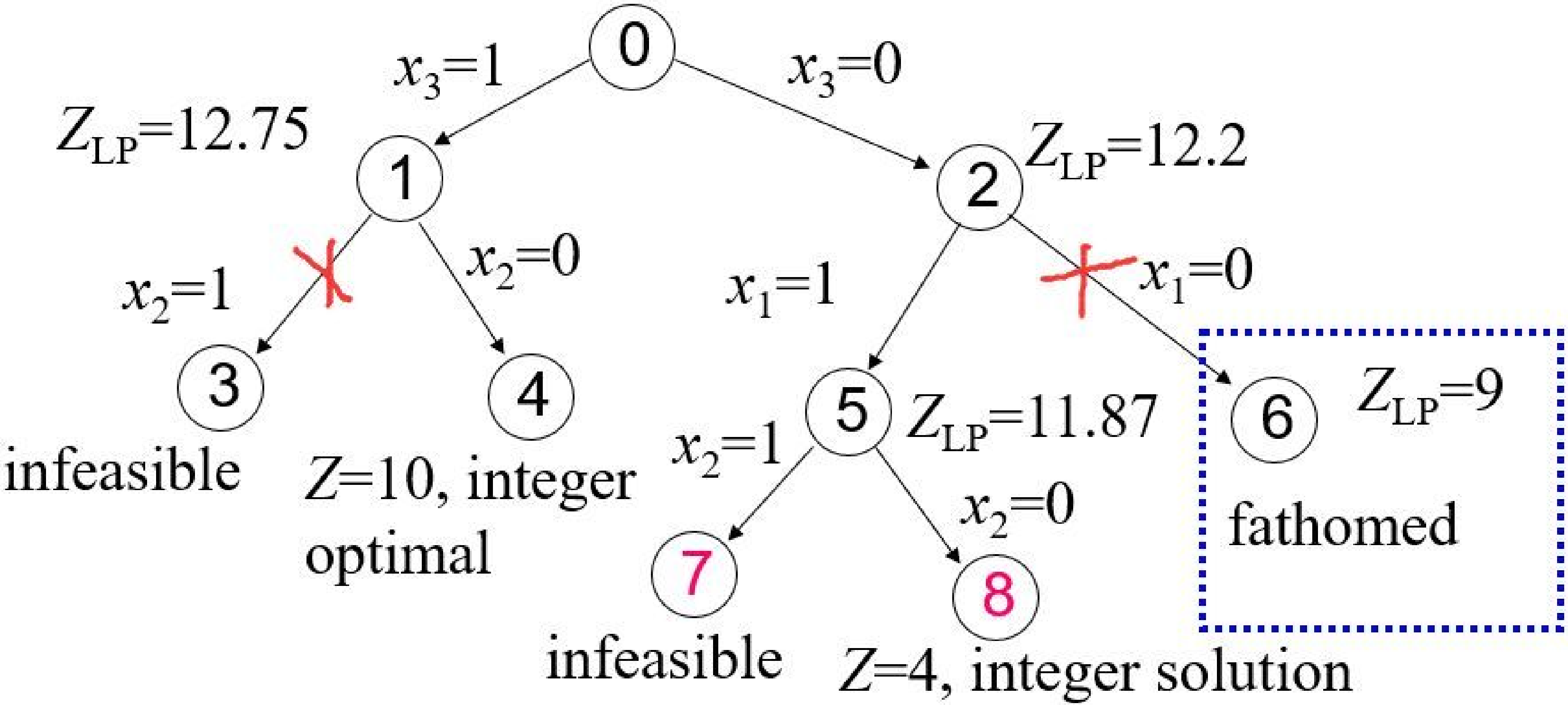
x_1, x_2 are binary variables

Maximize $4x_1+9x_2$

s.t. $5x_1+8x_2 \leq 12$

x_1, x_2 are binary variables

Branch and Bound - Example



Extension to General Problems

For problems with general integer constraint (not just binary constraints), the algorithm is virtually identical

Only difference is that we pick non-integer $\tilde{x}_j$ and then add Solve-Relaxation($S \cup \{x_i \geq \lceil \tilde{x}_j \rceil\}$), Solve-Relaxation ($S \cup \{x_i \leq \lfloor \tilde{x}_j \rfloor\}$)to the frontier

Can deal with mixed integer problems (some non-integer variables), by simply not branching on non-integer variables, and by resolving over non-integer variables after rounding integer variables