IE418: Integer Programming

Jeff Linderoth

Department of Industrial and Systems Engineering
Lehigh University

19th January 2005

Today's Outline

- About this class
  - About me
  - Say Cheese
  - Quiz Number 0
- Integer Programs
  - Why would you possibly care?
  - First Definitions
  - Modeling: Our first integer programs

Class Overview

- Meeting Times: Monday-Wednesday 5:30–6:50
- Office Hours:
  - Monday 8AM–9AM
  - Wednesday 7–8PM
  - By Appointment (8-4879).
- Course HomePage:
  - `http://www.lehigh.edu/~jtl3/teaching/ie418`
  - I will try to post (draft) outlines of lecture notes there before class.
- Syllabus dates are more than somewhat tentative

Course Details

- Learning is better if you participate.
  - I will call on you during class. (Gasp!)
  - Sitting in or Auditing (Why!)
- Six or Seven Problem Sets
  - Can work (and writeup) in groups of two.
  - The lowest score will count only 50% as much as the others in determining your homework average score.
  - Don't be late! 10% Grade penalty for every late day.
  - Most problem sets consist of computational assignment, so if you are computer illerate, you may want to pair with someone who is not.
- Mid Term Exam
  - Planning an in-class exam
- Final Exam
  - Planning a take-home exam

Grading

- I don't view grades in (elective) graduate courses as very important.
- You should be here because you want to be here, and you should learn because you want to learn.
- Nevertheless, they make me assign grades. Therefore...
  - 50% Problem Sets
  - 20% Mid Term Exam
  - 30% Final Exam

Course Topics

(Subject to Change)

Module #1: IP Basics

- Modeling
  - Modeling problems with integer decision variables
  - Formulating and solving MIPS with an AML
- Branch-and-Bound
  - The very basics of the "workhorse" algorithm for solving IPs
- Software for IP

Module #2: IP Theory

- Complexity Theory
  - What makes a problem "hard" or "easy"
  - The complexity classes $\mathcal{P}$, and $\mathcal{NP}$, and $\mathcal{NP}$-completeness
- Polyhedral Theory
  - Dimension, facets, polarity, and the equivalence of separation and optimization.

Course Topics (2)

Module #3: Branching-Based Algorithms

- Preprocessing and Probing
- Cutting Planes and lifting
- Branching and Node Selection Rules

Module #4: Advanced Topics

- Lagrangian Relaxations, Bender's Decomposition, Branch-and-Price.
- Primal and Lattice-Based methods

Course Objectives

1. Be able to read, digest, and understand scholarly journal articles on integer programming.
2. Become acquainted with "off the shelf" solvers for integer programs, learn what the algorithmic parameters mean, how to tune them for improved performance, and how to customize them for specialized algorithms.
3. Understand how integer variables are used for formulating complex mathematical models, and in particular, what makes one valid model "better" than another.
4. Know the basic concepts of complexity theory, to be able to understand the difference between an "easy" problem and a "hard" problem.

Course Objectives

5. Absorb basic polyhedral theory, and use this theory to describe properties of polyhedra arising from integer programs.
6. Understand the theory of valid inequalities and the lifting of valid inequalities.
7. Learn different classes of valid inequalities used in commercial solvers and also problem specific classes of valid inequalities.
8. Understand decomposition-based (problem specific) techniques for large-scale problems.

Great Expectations

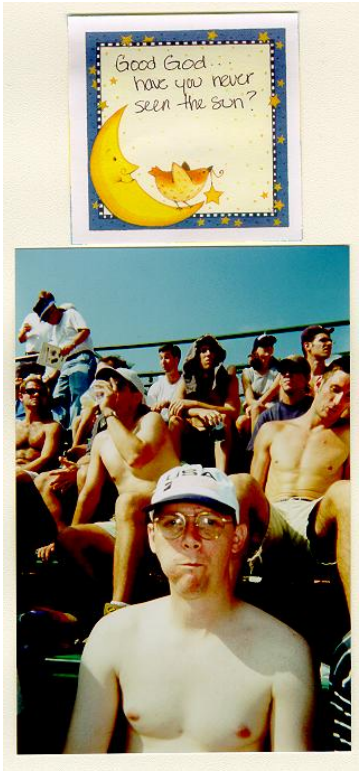
I am expected to...

- Teach
- Be at my office hours
- Give you feedback on how you are doing in a timely fashion

You are expected to...

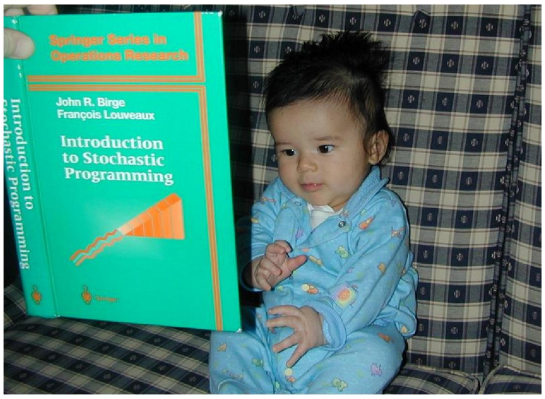
- Learn
- Attend lectures and participate
- Do the problem sets
- Not be rude, if possible.
  - Sleeping
  - Cell Phones
  - Leaving in the middle of lecture

About me...



- B.S. (G.E.), UIUC, 1992.
- M.S., OR, GA Tech, 1994.
- Ph.D., Optimization, GA Tech, 1998
- 1998-2000 : MCS, ANL
- 2000-2002 : Axioma, Inc.
- Research Areas: Large Scale Optimization, High Performance Computing.
- Married. One child, Jacob, born 10/28/02. He is awesome.
- Hobbies: Golf, Stochastic Programming, Chess.

Picture Time



Programming? I Hate Programming!

Question

What does "Programming" mean in "Mathematical Programming", "Linear Programming", "Stochastic Programming", "Integer Programming"?

Answer

Planning

- Mathematical Programming (Optimization) is about decision making.
- Integer Programming is about decision making *with integers*.
- More precisely, decision making in which some of the decisions make take only certain integer values.

Programming With Integers?!?!?!

- Why in the heck would we ever want to program with integers?
- If the variable is associated with a physical entity that is indivisible, then it must be integer.
  - Number of airplanes to produce
  - Number of ears of corn in the state of Illinois to harvest in the month of October.
- How are these two decisions different?
  - The point here is that *sometimes* a continuous approximation to the discrete (integer) decision is accurate enough for practical purposes.

IMPORTANT Programming with Integers

- We can use 0-1 (binary) variables for a variety of purposes.
  - Modeling yes/no decisions.
  - Enforcing disjunctions.
  - Enforcing logical conditions.
  - Modeling fixed costs.
  - Modeling piecewise linear functions.
- In most of these cases the continuous approximation to the discrete decision is *not* accurate enough for practical purposes.
- Before we get to specific modeling instance, let's look at the general model...

MIP

(Linear) Mixed-Integer Programming Problem: (MIP)

$$\max\{c^T x + h^T y \mid Ax + Gy \leq b, x \in \mathbb{Z}^n_+, y \in \mathbb{R}^p_+\}$$

- **Instance**: $c \in \mathbb{R}^n$, $h \in \mathbb{R}^p$, $A \in \mathbb{R}^{m \times n}$, $G \in \mathbb{R}^{m \times p}$

- MATH NOTATION QUIZ
  - What the heck is $\mathbb{R}^p_+$?
  - What the heck is $\mathcal{Z}^n_+$?
  - $\in$?

General Comments

- Note that the textbook considers maximization problems by default.
- Many people (myself included) often consider minimization as the default, so be ready to be confused
- One further assumption we will make, and never mention again, is that the input data is rational. e.g. $(c \in \mathbb{Q}^n)$
  - This is an important assumption since with irrational data, certain "intuitive" results no longer hold.
  - A computer can only understand rational data anyway, so this is not an unreasonable assumption.
  - Note that you implicitly assumed this in your linear programming course anyway!

How Hard is Integer Programming?

- Solving general integer programs can be much more difficult than solving linear programs.
- This is more than just an empirical statement
  - There is a whole theory surrounding it
  - You will learn some of the gory details
- Solving the associated linear programming relaxation results in an upper bound on the optimal solution to the MIP.
  - Rounding to a feasible integer solution may be difficult or impossible
  - The optimal solution to the LP relaxation can be arbitrarily far away from the optimal solution to the MIP.

Integer Programming Classes

- We call the problem *mixed* integer programming due to the presence of continuous variables
- In some problems x are allowed to take on values only 0 or 1
  - Such variables are called binary.
  - Integer programs involving *only* binary variables are called binary integer programs (BIPs). ($x \in \mathbb{B}^n$)
- Pure Integer Programming
  - G, h, y not present: $\max\{c^T x \mid Ax \leq b, x \in \mathbb{Z}^n_+\}$
- Mixed 0-1 Programming
  - $x \in \mathcal{B}^n$. G, h, y present: $\max\{c^T x + h^T y \mid Ax + Gy \leq b, x \in \mathbb{B}^n_+, y \in \mathbb{R}^p_+\}$
- Linear Programming(!)
  - c, A, x not present: $\max\{h^T y \mid Gy \leq b, y \in \mathbb{R}^p_+\}$

Combinatorial Optimization

- A combinatorial optimization problem $CP = (N, \mathcal{F})$ consists of
  - A finite ground set N,
  - A set $\mathcal{F} \subseteq 2^N$ of feasible solutions
  - Costs $c_j \ \forall j \in N$
- The cost of $F \in \mathcal{F}$ is $c(F) = \sum_{j \in F} c_j$.
- The combinatorial optimization problem is then

$$\max\{c(F) : F \in \mathcal{F}\}$$

- Many COPs can be written as BIPs or MIPs.
- We'll see an example soon...

The Knapsack Problem

- A burglar has a knapsack of size b. He breaks into a store that carries a set of items N. Each item has profit c_j and size a_j.
- What items should the burglar select in order to optimize his heist?

$$x_j = \begin{cases} 1 & \text{Item } j \text{ goes in the knapsack} \\ 0 & \text{Otherwise} \end{cases}$$

$$z_{\text{HEIST}} = \max\{\sum_{j\in N} c_j x_j : \sum_{j\in N} a_j x_j \leq b,\ x_j \in \{0,1\}\ \forall j \in N\}.$$

- Integer Knapsack Problem:

$$z_{\text{HEIST}} = \max\{\sum_{j\in N} c_j x_j : \sum_{j\in N} a_j x_j \leq b,\ x_j \in \mathbb{Z}_+\ \forall j \in N\}.$$

Couse Details
Motivation for IP
Integer Programs
Simple Binary Models
Combinatorial Optimization Problems
Special Ordered Sets
"Algorithmic" Modeling

Fall into the...

- Given m machines and n jobs, find a least cost assignment of jobs to machines not exceeding the machine capacities
- Each job j requires a_{ij} units of machine i's capacity b_i

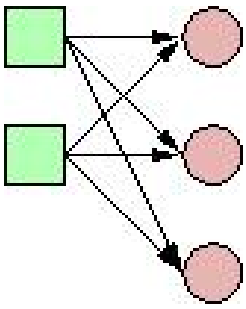
$$\begin{aligned} \min z &\equiv \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \text{s.t.} \sum_{j=1}^{n} a_{ij}x_{ij} &\leq b_i \quad \forall i \qquad \text{(Machine Capacity)} \\ \sum_{i=1}^{m} x_{ij} &= 1 \quad \forall j \qquad \text{(Assign all jobs)} \\ x_{ij} &\in \{0,1\} \quad \forall i,j \end{aligned}$$

Modeling Dependent Decisions

- We can also use binary variables to enforce the condition that a certain action can only be taken if some other action is also taken.
- Suppose x and y are variables representing whether or not to take certain actions.
- The constraint $x \leq y$ says "only take action x if action y is also taken"

- Uncapacitated facility location.
  - Set of facilities J, Set of customers I. Which facilities should be opened in order to meet customer demand at minimum cost?

UFL



- $x_j = 1$: If and only if we open facility j
- y_{ij}: Fraction of customer j's demand satisfied by facility i

$$\min \sum_{j \in J} c_j x_j + \sum_{i \in I} \sum_{j \in J} f_{ij} y_{ij}$$

$$\begin{aligned}
\sum_{j \in N} y_{ij} &= 1 \quad \forall i \in I \\
\sum_{i \in I} y_{ij} &\leq x_j \quad \forall j \in J && (1) \\
x_j &\in \{0,1\} \quad \forall j \in J && (2) \\
y_{ij} &\geq 0 \quad \forall i \in I, \forall j \in J && (3)
\end{aligned}$$

Selecting from a Set

- We can use constraints of the form $\sum_{j \in T} x_j \geq 1$ to represent that at least one item should be chosen from a set T.
  - Similarly, we can also model that at most one or exactly one item should be chosen.
- Example: Set covering problem
- If A in a 0-1 matrix, then a set covering problem is any problem of the form

$$\begin{array}{ll} \min & c^T x \\ s.t. & Ax \geq e^1 \\ & x_j \in \{0,1\} \quad \forall j \end{array}$$

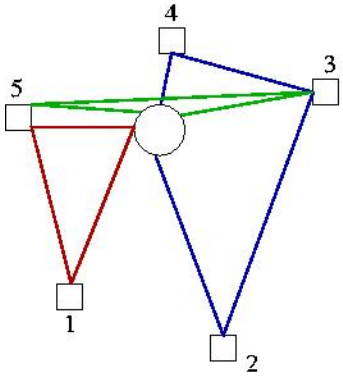
- Set Packing: $Ax \leq e$
- Set Partitioning: $Ax = e$

[1] It is common to denote the vector of 1's as e

COP—Set {Covering, Packing, Partitioning}

- A combinatorial optimization problem $CP = (N, \mathcal{F})$ consists of
  - A finite ground set N,
  - A set $\mathcal{F} \subseteq 2^N$ of feasible solutions, and
  - A cost function c
- Each row of A represents an item from N
- Each column A_j represents a subset $N_j \in \mathcal{F}$ of the items.
- Each variable x_j represents selecting subset N_j.
- The constraints say that $\cup_{\{j|x_j=1\}} N_j = N$.
- In other words, each item must appear in at least one selected subset.

Vehicle Routing



	x_1	x_2	x_3	$\cdots$		
Customer 1 :	1	0	0	$\vdots$	$=$	1
Customer 2 :	0	1	0	$\vdots$	$=$	1
Customer 3 :	0	1	1	$\vdots$	$=$	1
Customer 4 :	0	1	0	$\vdots$	$=$	1
Customer 5 :	1	0	1	$\vdots$	$=$	1

- This is a very flexible modeling trick
- You can list all feasible routes, allowing you to handle "weird" constraints like time windows, strange precedence rules, nonlinear

The Farmer's Daughter

- This is *The Most Famous Problem in Combinatorial Optimization!*
- A traveling salesman must visit all his cities at minimum cost.

$$\min \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij}$$

- Given directed (complete) graph with node set N
- Costs c_{ij} of traveling from city i to city j
- $x_{ij} = 1$ if and only if salesman goes from city i to city j

$$\sum_{i \in N} x_{ij} = 1 \quad \forall j \in N$$

$$\sum_{j \in N} x_{ij} = 1 \quad \forall i \in N$$

Subtour elimination constraint:

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 1 \quad \forall S \subseteq N, 2 \leq |S| \leq |N| - 2$$

TSP Trivia Time!

- Historically, this was the first problem on which "branch and cut" was performed (by Dantzig, Fulkerson, and Johnson).

$$101851798816724304313422284420468908052573419683296$$
$$8125318070224677190649881668353091698688.$$

- Is this...
  a) The number of times an undergraduate student asked me where room 355 was this week?
  b) The number of subatomic particles in the universe?
  c) The number of subtour elimination constraints when $|N| = 299$.
  d) All of the above?
  e) None of the above?

Couse Details
Motivation for IP
Integer Programs
Simple Binary Models
Combinatorial Optimization Problems
Special Ordered Sets
"Algorithmic" Modeling

Answer Time

- The answer is (e). (a)–(c) are all too small (as far as I know) :-). (It is (c), for $|N| = 300$).
- "Exponential" is **really** big.
- Yet people have solved TSP's with $|N| > 16,000$!
- You will learn how to solve these problems too!
- The "trick" is to only add the subset of constraints that are necessary to prove optimality. branch-and-cut

Modeling a Restricted Set of Values

- We may want variable x to only take on values in the set $\{a_1, \ldots, a_m\}$.
- We introduce m binary variables $y_j, j = 1, \ldots, m$ and the constraints

$$x = \sum_{j=1}^{m} a_j y_j,$$
$$\sum_{j=1}^{m} y_j = 1,$$
$$y_j \in \{0, 1\}$$

- This is called a *special ordered set* (SOS) of variables.

Example—Building a warehouse

- Suppose we are modeling a facility location problem in which we must decide on the size of a warehouse to build.
- The choices of sizes and their associated cost are shown below:

Size	Cost
10	100
20	180
40	320
60	450
80	600

Warehouse sizes and costs

Warehouse Modeling

- Using binary decision variables $x_1, x_2, \ldots, x_5$, we can model the cost of building the warehouse as

$$\text{COST} \equiv 100x_1 + 180x_2 + 320x_3 + 450x_4 + 600x_5.$$

The warehouse will have size

$$\text{SIZE} \equiv 10x_1 + 20x_2 + 40x_3 + 60x_4 + 80x_5,$$

and we have the SOS constraint

$$x_1 + x_2 + x_3 + x_4 + x_5 = 1.$$

Piecewise Linear Cost Functions

- We can use binary variables to model arbitrary piecewise linear cost functions.
- The function is specified by ordered pairs $(a_i, f(a_i))$ and we wish to evaluate $f(x)$
- We have a binary variable y_i, which indicates whether $a_i \leq x \leq a_{i+1}$.
- To evaluate the function, we will take linear combinations $\sum_{i=1}^{k} \lambda_i f(a_i)$ of the given functions values.
- This only works if the only two nonzero $\lambda_i's$ are the ones corresponding to the endpoints of the interval in which x lies.

Minimizing Piecewise Linear Cost Functions

- The following formulation minimizes the function.

$$\begin{array}{ll} \min & \sum_{i=1}^{k} \lambda_i f(a_i) \\ s.t. & \sum_{i=1}^{k} \lambda_i = 1, \\ & \lambda_1 \leq y_1, \\ & \lambda_i \leq y_{i-1} + y_i, \qquad i \in [2..k-1], \\ & \lambda_k \leq y_{k-1}, \\ & \sum_{i=1}^{k-1} y_i = 1, \\ & \lambda_i \geq 0, \\ & y_i \in \{0,1\}. \end{array}$$

- The key is that if $y_j = 1$, then $\lambda_i = 0,\ \forall i \neq j, j+1$.

SOS2

- A "better" formulation involves the use of *special ordered sets of type 2*
  - SOS2 : A set of variables of which at most two can be positive. If two are positive, they must be adjacent in the set.

$$\begin{array}{ll} \min & \sum_{i=1}^{k} \lambda_i f(a_i) \\ s.t. & \sum_{i=1}^{k} \lambda_i = 1, \\ & \lambda_i \geq 0, \\ & \{\lambda_1, \lambda_2, \ldots, \lambda_k\} \quad \text{SOS2} \end{array}$$

- The adjacency conditions of SOS2 are enforced by the solution algorithm
- (All) commercial solvers allow you to specify SOS2

Modeling Disjunctive Constraints

- We are given two constraints $a^T x \geq b$ and $c^T x \geq d$ with nonnegative coefficients.
- Instead of insisting both constraints be satisfied, we want at least one of the two constraints to be satisfied.
- To model this, we define a binary variable y and impose

$$\begin{aligned} a^T x &\geq yb, \\ c^T x &\geq (1-y)d, \\ y &\in \{0,1\}. \end{aligned}$$

Couse Details
Motivation for IP
Integer Programs
Simple Binary Models
Combinatorial Optimization Problems
Special Ordered Sets
"Algorithmic" Modeling

Modeling Disjunctive Constraints

- More generally, we can impose that exactly k out of m constraints be satisfied with

$$\begin{aligned} (a_i')^T x &\geq b_i y_i, \quad i \in [1..m] \\ \sum_{i=1}^m y_i &\geq k, \\ y_i &\in \{0,1\} \end{aligned}$$

The Bag of Tricks

- Indicator variables (Positivity of variables)
  - Limiting the Number of Positive Variables
  - "Fixed Charge" problems
  - Minimum production level
- Indicator variables (Validity of constraints)
  - Either-or
  - If-then
  - k out of n

Couse Details
Motivation for IP
Integer Programs
Simple Binary Models
Combinatorial Optimization Problems
Special Ordered Sets
"Algorithmic" Modeling

The Bag of Tricks

- Special Ordered Sets
- Nonconvex regions
- Economies of Scale
- Discrete Capacity Extensions
- Maximax or Minimin

The Slide of Tricks. Indicator Variables...

- $\delta = 1 \Rightarrow \sum_{j \in N} a_j x_j \leq b$
  - $\sum_{j \in N} a_j x_j + M\delta \leq M + b$
- $\sum_{j \in N} a_j x_j \leq b \Rightarrow \delta = 1$
  - $\sum_{j \in N} a_j x_j - (m - \epsilon)\delta \geq b + \epsilon$
- $\delta = 1 \Rightarrow \sum_{j \in N} a_j x_j \geq b$
  - $\sum_{j \in N} a_j x_j + m\delta \geq m + b$
- $\sum_{j \in N} a_j x_j \geq b \Rightarrow \delta = 1$
  - $\sum_{j \in N} a_j x_j - (M + \epsilon)\delta \leq b - \epsilon$

Definitions

- δ: Indicator variable ($\delta \in \{0, 1\}$).
- M: Upper bound on $\sum_{j \in N} a_j x_j - b$
- m: Lower bound on $\sum_{j \in N} a_j x_j - b$
- ϵ: Small tolerance beyond which we regard the constraint as haven been broken.
  - If $a_j \in \mathbb{Z}$, $x_j \in \mathbb{Z}$, then we can take $\epsilon = 1$.

Couse Details
Motivation for IP
Integer Programs
Simple Binary Models
Combinatorial Optimization Problems
Special Ordered Sets
"Algorithmic" Modeling

Modeling Trick #1

- Indicating Constraint (Non)violation
- Suppose we wish to indicate whether or not an inequality $\sum_{j \in N} a_j x_j \leq b$ holds by means of an indicator variable δ. That is.

$$\boxed{\delta = 1 \Rightarrow \sum_{j \in N} a_j x_j \leq b}$$

- This can be represented by the constraint
  - $\sum_{j \in N} a_j x_j + M\delta \leq M + b$

Trick #1... The Logic

$$\boxed{\delta = 1 \Rightarrow \sum_{j \in N} a_j x_j \leq b \Leftrightarrow \sum_{j \in N} a_j x_j + M\delta \leq M + b}$$

- (Thinking)...
  - $\delta = 1 \Rightarrow \sum_{j \in N} a_j x_j - b \leq 0$
  - $1 - \delta = 0 \Rightarrow \sum_{j \in N} a_j x_j - b \leq 0$
  - $\sum_{j \in N} a_j x_j - b \leq M(1 - \delta)$
- Does it work?
  - $\delta = 0 \Rightarrow \sum_{j \in N} a_j x_j - b \leq M$
    - (true by definition of M)
  - $\delta = 1 \Rightarrow \sum_{j \in N} a_j x_j - b \leq 0$

Modeling Trick #2

$$\boxed{\sum_{j \in N} a_j x_j \leq b \Rightarrow \delta = 1}$$

- $\delta = 0 \Rightarrow \sum_{j \in N} a_j x_j \not\leq b$
- $\delta = 0 \Rightarrow \sum_{j \in N} a_j x_j > b$
- $\delta = 0 \Rightarrow \sum_{j \in N} a_j x_j \geq b + \epsilon$
  - If a_j, x_j are integer, we can choose $\epsilon = 1$
- Model as $\sum_{j \in N} a_j x_j - (m - \epsilon)\delta \geq b + \epsilon$
  - m is a lower bound for the expression $\sum_{j \in N} a_j x_j - b$

Some Last Modeling Tricks

$$\boxed{\delta = 1 \Rightarrow \sum_{j \in N} a_j x_j \geq b}$$

- Model as $\sum_{j \in N} a_j x_j + m\delta \geq m + b$

$$\boxed{\sum_{j \in N} a_j x_j \geq b \Rightarrow \delta = 1}$$

- Model as $\sum_{j \in N} a_j x_j - (M + \epsilon)\delta \leq b - \epsilon$
- You can obtain these by just transforming the constraints to $\leq$ form and using the first two tricks.

Example

- Use a 0-1 variable δ to indicate whether *or not* the constraint $2x_1 + 3x_2 \leq 1$ is satisfied.
  - x_1, x_2 are nonnegative continuous variables that cannot exceed 1.
- $\delta = 1 \Leftrightarrow 2x_1 + 3x_2 \leq 1$
- M : Upper Bound on $2x_1 + 3x_2 - 1$. 4 works
- m : Lower Bound on $2x_1 + 3x_2 - 1$. -1 works.
- ϵ : 0.1

Example, Cont.

- ($\Rightarrow$) Recall the trick.
  - $\delta = 1 \Rightarrow \sum_{j \in N} a_j x_j \leq b \Leftrightarrow \sum_{j \in N} a_j x_j + M\delta \leq M + b$
- $2x_1 + 3x_2 + 4\delta \leq 5$
- ($\Leftarrow$). Recall the trick.
  - $\sum_{j \in N} a_j x_j \leq b \Rightarrow \delta = 1 \Leftrightarrow \sum_{j \in N} a_j x_j - (m - \epsilon)\delta \geq b + \epsilon$
- $2x_1 + 3x_2 + 1.1\delta \geq 1.1$

$$\begin{aligned} 2x_1 + 3x_2 + 4\delta &\leq 5 \\ 2x_1 + 3x_2 + 1.1\delta &\geq 1.1 \end{aligned}$$

A More Realistic Example

- PPP—Production Planning Problem. (A simple linear program).
- An engineering plant can produce five types of products: $p_1, p_2, \ldots p_5$ by using two production processes: grinding and drilling. Each product requires the following number of hours of each process, and contributes the following amount (in hundreds of dollars) to the net total profit.

	p_1	p_2	p_3	p_4	p_5
Grinding	12	20	0	25	15
Drilling	10	8	16	0	0
Profit	55	60	35	40	20

PPP – More Info

- Each unit of each product take 20 manhours for final assembly.
- The factory has three grinding machines and two drilling machines.
- The factory works a six day week with two shifts of 8 hours/day. Eight workers are employed in assembly, each working one shift per day.

PPP

maximize

$$55x_1 + 60x_2 + 35x_3 + 40x_4 + 20x_5 \qquad \text{(Profit/week)}$$

subject to

$$\begin{aligned}
12x_1 + 20x_2 + 0x_3 + 25x_4 + 15x_5 &\leq 288 && \text{(Grinding)} \\
10x_1 + 8x_2 + 16x_3 + 0x_4 + 0x_5 &\leq 192 && \text{(Drilling)} \\
20x_1 + 20x_2 + 20x_3 + 20x_4 + 20x_5 &\leq 384 && \text{Final Assembly} \\
x_i &\geq 0 && \forall i = 1, 2, \ldots 5
\end{aligned}$$

Another PPP Modeling Example

- Let's model the following situation.
  - If we manufacture $P1$ or $P2$ (or both), then at least one of $P3$, $P4$, $P5$ must also be manufactured.
- Recall – We have indicator variables z_j that indicate when each of the $x_j > 0$.
  - How do we model $x_j > 0 \Rightarrow z_j = 1$?.

Modeling the Logic

Answer: $x_j \leq M z_j$

- Given that we have included the constraints $x_j \leq M z_j$, we'd like to model the following implication:
  - $z_1 + z_2 \geq 1 \Rightarrow z_3 + z_4 + z_5 \geq 1$
- Can you just "see" the answer?
- I can't. So let's try our "formulaic" approach.
- Think of it in two steps
  - $z_1 + z_2 \geq 1 \Rightarrow \delta = 1$
  - $\delta = 1 \Rightarrow z_3 + z_4 + z_5 \geq 1$.

Look up the Tricks

- First we model the following:
  - $z_1 + z_2 \geq 1 \Rightarrow \delta = 1$
- The formula from the bag o' tricks
- $\sum_{j \in N} a_j x_j \geq b \Rightarrow \delta = 1 \Leftrightarrow \sum_{j \in N} a_j x_j - (M + \epsilon)\delta \leq b - \epsilon$
- M : Upper Bound on $\sum_{j \in N} a_j z_j - b$
  - $M = 1$ in this case. $(z_1 \leq 1, z_2 \leq 1, b = 1)$.
- ϵ : "Tolerance Level" indicating the minimum about by which the constraint can be violated.
  - $\epsilon = 1$ in this case!
  - If the constraint is going to be violated, then it will be violated by at least one.

Modeling $z_1 + z_2 \geq 1 \Rightarrow \delta = 1$, Cont.

- Just plug in the formula $\sum_{j \in N} a_j x_j - (M + \epsilon)\delta \leq b - \epsilon$
  - $z_1 + z_2 - 2\delta \leq 0$
- Does this do what we want?

z_1	z_2	δ
0	0	≥ 0
0	1	$\geq 1/2 (\Rightarrow = 1)$
1	0	$\geq 1/2 (\Rightarrow = 1)$
1	1	≥ 1

Second Part

- Want to model the following:
  - $\delta = 1 \Rightarrow z_3 + z_4 + z_5 \geq 1$.
- The formula from the bag o' tricks
- $\delta = 1 \Rightarrow \sum_{j \in N} a_j x_j \geq b \Leftrightarrow \sum_{j \in N} a_j x_j + m\delta \geq m + b$
- m : lower bound on $\sum_{j \in N} a_j x_j - b$.
  - $m = -1$. $(z_1 \geq 0, z_2 \geq 0, b = 1)$.
- Plug in the formula:
  - $z_3 + z_4 + z_5 - \delta \geq 0$
- It works! (Check for $\delta = 0, \delta = 1$).

PPP, Make 1 or 2 $\Rightarrow$ make 3, 4, or 5

maximize

$$55x_1 + 60x_2 + 35x_3 + 40x_4 + 20x_5 \qquad \text{(Profit/week)}$$

subject to

$$\begin{aligned}
12x_1 + 20x_2 + 0x_3 + 25x_4 + 15x_5 &\leq 288 \\
10x_1 + 8x_2 + 16x_3 + 0x_4 + 0x_5 &\leq 192 \\
20x_1 + 20x_2 + 20x_3 + 20x_4 + 20x_5 &\leq 384 \\
x_i &\leq M_i z_i \qquad \forall i = 1, 2, \ldots 5 \\
z_1 + z_2 - 2\delta &\leq 0 \\
z_3 + z_4 + z_5 - \delta &\geq 0 \\
x_i &\geq 0 \qquad \forall i = 1, 2, \ldots 5 \\
z_i &\in \{0, 1\} \forall i = 1, 2, \ldots 5 \\
\delta &\in \{0, 1\}
\end{aligned}$$

Cool Things You Can Now Do

- Either constraint 1 or constraint 2 must hold
  - Create indicators δ_1, δ_2, then $\delta_1 + \delta_2 \geq 1$
- At least one constraint of all the constraints in M should hold
  - $\sum_{i \in M} \delta_i \geq 1$
- At least k of the constraints in M must hold
  - $\sum_{i \in M} \delta_i \geq k$
- If x, then y
  - $\delta_y \geq \delta_x$