CO250 Spring 2020

Jacky Zhao

June 8, 2020

1 Introduction

1.1 Abstract Optimization Problem

An *abstract optimization problem (P)* is of the following form:

- Given: a set $\mathbf{A} \subseteq \mathbb{R}^n$ and a function $f : \mathbf{A} \to \mathbb{R}$
- Goal: find $x \in \mathbf{A}$ that minimizes/maximizes f
- Bad news: Hard to solve & may not be well defined

We look at 3 special cases of (P) in this course:

1. Linear Programming (LP)
    - $\mathbf{A}$ is simply given by *linear* constrains, and f is a *linear* function
2. Integer Programming (IP)
    - Same as above, but now we want max/min over the *integer* points in $\mathbf{A}$
3. Non-linear Programming (NLP)
    - $\mathbf{A}$ is given by *non-linear* constrains, and f is a *non-linear* function

1.1.1 Example: Water Tech

WaterTech produces 4 products, $P = \{1, 2, 3, 4\}$, from the following resources:

- time on two machines
- skilled and unskilled labour

The following table gives precise requirements:

Product	Machine 1	Machine 2	Skilled Labour	Unskilled Labour	Unit Sale Price
1	11	4	8	7	300
2	7	6	5	8	260
3	6	5	5	7	220
4	5	4	6	4	180

Restrictions:

- WaterTech has 700h on machine 1 and 500h on machine 2 available
- it can purchase 600h of skilled labour at \$8 per hour and at most 650h of unskilled labour at \$6 per hour

Question:
How much of each product should WaterTech produce in order to maximize profit?

1.2 Ingredients of a math model:

- Decision variables: Capture unknown information
- Constraints: Describe which assignments to variables are feasible
- Objective function: A function of the variables that we would like to maximize/minimize

1.3 Variables

WaterTech needs to decide how many units of each product to produce, so introduce some variables:

- x_i for number of labour to purchase
- y_s, y_u for number of hours of skilled/unskilled labour to purchase

1.4 Constrains

What makes an assignment to $\{x_i\} \in P, y_s, y_u$ a feasible assignment?

For example, a production plan described by an assignment may not use more than 700h of time on machine 1

$$11x_1 + 7x_2 + 6x_3 + 5x_4 \leq 700$$

Similarly, we may not use more than 500h of machine 2 time

$$4x_1 + 6x_2 + 5x_3 + 4x_4 \leq 500$$

Producing x_i units of product $i \in P$ must require less than y_s units of skilled labour

$$8x_1 + 5x_2 + 5x_3 + 6x_4 \leq y_s$$

Similar story for unskilled labour:

$$7x_1 + 8x_2 + 7x_3 + 5x_4 \leq y_u$$

Since amount of labour that can be purchased is limited, we also have

$$y_s \leq 600$$
$$y_u \leq 650$$

1.5 Objective Function

Revenue from sales:

$$300x_1 + 260x_2 + 220x_3 + 180x_4$$

Cost of labour:

$$8y_s + 6y_u$$

Objective Function:

$$\text{maximize } 300x_1 + 260x_2 + 220x_3 + 180x_4 - 8y_s - 6y_u$$

The complete model for WaterTech problem is:

$$\begin{aligned}
\max \quad & 300x_1 + 260x_2 + 220x_3 + 180x_4 - 8y_s - 6y_u \\
\text{s.t} \quad & 11x_1 + 7x_2 + 6x_3 + 5x_4 \leq 700 \\
& 4x_1 + 6x_2 + 5x_3 + 4x_4 \leq 500 \\
& 8x_1 + 5x_2 + 5x_3 + 6x_4 \leq y_s \\
& 7x_1 + 8x_2 + 7x_3 + 5x_4 \leq y_u \\
& y_s \leq 600 \\
& y_u \leq 650 \\
& x_1, x_2, x_3, x_4, y_u, y_s \geq 0
\end{aligned}$$

Solution obtained via CPLEX is:

$$\begin{aligned}
x &= (16 + \frac{2}{3}, 50, 0, 33 + \frac{1}{3})^T \\
y_s &= 583 + \frac{1}{3} \\
y_u &= 650 \\
Profit &= 15433 + \frac{1}{3}
\end{aligned}$$

Notice that the solution is fractional, which may or may not be correct depending on the question

1.6 Correctness of Model

First, define some terminologies:

- Word description of problem
  - Similarly, a solution to the word description is an assignment to the unknowns
- Formulation
  - A solution to the formulation is an assignment to all of its variables

A solution feasible if all constrains are satisfied, optimal if no other feasible solution exists

One way to show correctness is to define a mapping between feasible solutions to the word description, and feasible solutions to the model, and vice versa.

2 Linear Program Model (LP)

2.1 Linear Functions

Affine Functions

- A function $f : \mathbb{R}^n \to \mathbb{R}$ is affine if $f(x) = \alpha^T x + \beta$ for $\alpha \in \mathbb{R}^n, \beta \in \mathbb{R}$

Linear Functions

- An affine function with $\beta = 0$

2.2 Linear Program

Linear Program

- the optimization problem

$$max/min\{f(x) : f_i(x) \leq b_i, \forall 1 \leq i \leq m, x \in \mathbb{R}^n\}$$

  is a linear program if f is affine and $g_1, ..., g_m$ is finite number of linear functions

Some notes:

- dividing by variables is not allowed in LP
- can NOT have strict inequalities
- must have FINITE number of constraints

Example:

$$\begin{array}{ll} \max & \frac{-1}{x_1} - x_3 \\ \text{s.t.} & 2x_1 + x_3 < 3 \\ & x_1 + \alpha x_2 = 2 \qquad \forall \alpha \in \mathbb{R} \end{array}$$

Going back to the WaterTech problem, the model we created was in fact a linear program!

2.3 LP Models: Multiperiod Models

A multiperiod model is a problem where:

- time is split into periods
- we have to make a decision in each period
- all decisions influences the final outcome

Example:
KW Oil is a local supplier of heating oil, it needs to decide how much oil to purchase in order to satisfy demand of its customers.

Month	1	2	3	4
Demand(l)	5000	8000	9000	6000
Price(\$/l)	0.75	0.72	0.92	0.90

Question: When should we purchase how much oil when the goal is to min overall total cost?
Additional Complication: The company has a storage tank that

- has a capacity of 4000 litres of oil
- currently (beginning of month 1) contains 2000 litres of oil

Assumption: Oil is delivered at the beginning of the month, and consumption occurs int he middle of the month

Variables

- Need to decide how many litres of oil to purchase in each month i
  - make variable p_i for $i \in [4]$
- How much oil is stored in the tank at the beginning of month i?
  - make variable t_i for $i \in [4]$

Objective Function

Minimize cost of oil purchased

$$min \qquad 0.75p_1 + 0.72p_2 + 0.92p_3 + 0.90p_4$$

Constrains

We need

$$p_i + t_i \geq \text{(demand in month } i\text{)}$$

Balancing equation we get

$$p_i + t_i = \text{(demand in month } i\text{)} + t_{i+1}$$

So we have the following four constrains

$$\begin{aligned} p_1 + 2000 &= 5000 + t_2 \\ p_2 + t2 &= 5000 + t_3 \\ p_3 + t3 &= 5000 + t_4 \\ p_4 + t4 &\geq 6000 \end{aligned}$$

Complete LP for KW Oil

$$
\begin{aligned}
\min \quad & 0.75p_1 + 0.72p_2 + 0.92p_3 + 0.90p_4 \\
\text{s.t.} \quad & p_1 + 2000 = 5000 + t_2 \\
& p_2 + t2 = 5000 + t_3 \\
& p_3 + t3 = 5000 + t_4 \\
& p_4 + t4 \geq 6000 \\
& t_1 = 2000 \\
& t_i \leq 4000 \qquad (i = 2, 3, 4) \\
& t_1, p_i \geq 0 \qquad (i = 1, 2, 3, 4)
\end{aligned}
$$

Solving the LP gives the solution:
$p = (3000, 12000, 5000, 6000)^T$
$t = (2000, 0, 4000, 0)^T$

3 Integer Program (IP)

Recall the WaterTech problem

$$\begin{aligned}
\max \quad & 300x_1 + 260x_2 + 220x_3 + 180x_4 - 8y_s - 6y_u \\
\text{s.t} \quad & 11x_1 + 7x_2 + 6x_3 + 5x_4 \leq 700 \\
& 4x_1 + 6x_2 + 5x_3 + 4x_4 \leq 500 \\
& 8x_1 + 5x_2 + 5x_3 + 6x_4 \leq y_s \\
& 7x_1 + 8x_2 + 7x_3 + 5x_4 \leq y_u \\
& y_s \leq 600 \\
& y_u \leq 650 \\
& x_1, x_2, x_3, x_4, y_u, y_s \geq 0
\end{aligned}$$

$$\begin{aligned}
x &= (16 + \frac{2}{3}, 50, 0, 33 + \frac{1}{3})^T \\
y_s &= 583 + \frac{1}{3} \\
y_u &= 650 \\
Profit &= 15433 + \frac{1}{3}
\end{aligned}$$

Fractional solutions are often not desirable! Can we force the solution to be integer?

Integer Program

- an integer program is a linear program with added integrality constraints for some/all the variables
- we call an IP mixed if there are integer and fractional variables, and pure otherwise
- the difference between LPs and IPs is subtle, but LPs are easy to solve, IPs are not!

Integer program is provably difficult to solve!

- An algorithm is efficient if its running can be bounded by a polynomial of the input size of the instance
- LPs can be solved efficiently
- IPs are very unlikely to have efficient algorithms!

3.1 IP Models: Knapsack

Example:
KitchTech Shipping is a company wishes to ship crates from Toronto to Kitchener. Each crate type has a weight and value, and the total weight of crates shipped must not exceed 10,000 lbs.
Goal: Maximize the total value of shipped goods.

Type	1	2	3	4	5	6
weight (lbs)	30	20	30	90	30	70
value ($)	60	70	40	70	20	90

Variables:
One variable x_i for the number of crates of type i to pack.

Constraints:
The total weight of crates picked must not exceed 10000 lbs.

$$30x_1 + 20x_2 + 30x_3 + 90x_4 + 30x_5 + 70x_6 \leq 10000$$

Objective function:
Maximize the total value

$$max \qquad 60x_1 + 70x_2 + 40x_3 + 70x_4 + 20x_5 + 90x_6$$

Complete IP model for KitchTech Shipping:

$$\begin{aligned} \max \quad & 60x_1 + 70x_2 + 40x_3 + 70x_4 + 20x_5 + 90x_6 \\ \text{s.t.} \quad & 30x_1 + 20x_2 + 30x_3 + 90x_4 + 30x_5 + 70x_6 \leq 10000 \\ & x_i \geq 0 \qquad (i \in [6]) \\ & x_i \text{ integer } (i \in [6]) \end{aligned}$$

Let's make this shit more complicated with more rules...
Suppose that:

1. we must not send more than 10 crates of the same type
2. we can only send crates of type 3, if we send at least 1 crate of type 4

Note that we can send at most 10 crates of type 3 by the previous constraints!
By adding the following constraint, the added requirements is fulfilled:

$$x_3 \leq 10x_4$$

proving correctness of the added constraint:

- $x_4 \geq 1 \rightarrow$ new constraint is redundant
- $x_4 = 0 \rightarrow$ new constraint becomes $x_3 \leq 0$

Suppose we add another rule where we must:

1. take a total of at least 4 crates of type 1 or 2, or
2. take at least 4 crates of type 5 or 6

strategy:
Create a new variable y such that:

- $y = 1 \rightarrow x_1 + x_2 \geq 4$
- $y = 0 \rightarrow x_5 + x_6 \geq 4$
- and force y to take on value 0 or 1

So we add the following constraints:

- $x_1 + x_2 \geq 4y$
- $x_5 + x_6 \geq 4(1 - y)$
- $0 \leq y \leq 1$
- y integer

The variable y we added is called a binary variable. These are very useful for modelling logical constraints of the form:

- Condition (A or B) and C $\rightarrow$ D

So the finalized model would be:

$$
\begin{aligned}
\max \quad & 60x_1 + 70x_2 + 40x_3 + 70x_4 + 20x_5 + 90x_6 \\
\text{s.t.} \quad & 30x_1 + 20x_2 + 30x_3 + 90x_4 + 30x_5 + 70x_6 \leq 10000 \\
& x_3 \leq 10x_4 \\
& x_1 + x_2 \geq 4y \\
& x_5 + x_6 \geq 4(1 - y) \\
& x_i \geq 0 \qquad (i \in [6]) \\
& 0 \leq y \leq 1 \\
& \text{y integer} \\
& x_i \text{ integer } (i \in [6])
\end{aligned}
$$

3.2 IP Models: Scheduling

Example:
The neighborhood coffee shop is open on workdays. The daily demand for workers is given in the table. Each worker works for 4 consecutive days and has one day off.
Goal: Hire the smallest number of workers so that the demand can be met

Mon	Tues	Wed	Thurs	Fri
3	5	9	2	7

Variables:
Introduce variable x_d for every $d \in \{M, T, W, Th, F\}$ counting the number of people to hire with starting day d

Objective function:
Minimize the total number of people hired:

$$min \qquad x_M + x_T + x_W + x_{Th} + x_F$$

Constraints:
We need to ensure that enough people work on each of the days.
Question: Given a solution, how many people work on Monday?
Answer: All but those that start on Tuesday, i.e.

$$x_M + x_W + x_{Th} + x_F$$

And it must be greater than or equal to the number of workers required
So the complete LP is:

$$\begin{aligned} \min \quad & x_M + x_T + x_W + x_{Th} + x_F \\ \text{s.t.} \quad & x_M + x_W + x_{Th} + x_F \geq 3 \\ & x_M + x_T + x_{Th} + x_F \geq 5 \\ & x_M + x_T + x_W + x_F \geq 9 \\ & x_M + x_T + x_W + x_{Th} \geq 2 \\ & x_T + x_W + x_{Th} + x_F \geq 7 \\ & x \geq 0, x \text{ integer} \end{aligned}$$

4 Optimization on graphs

4.1 Graph Theory 101:

A graph G consist of:

- vertices $u, w, ... \in V$ (circles)
- edges $uw, wz, ... \in E$ (lines connecting circles)

Two vertices u and v are adjacent if $uv \in E$.
Vertices u and v are the endpoints of edge $uv \in E$
An edge $e \in E$ is incident to $u \in V$ if u is an endpoint of e.

An $s, t - path$ in $G = (V, E)$ is a sequence

$$v_1v_2, v_2v_3, v_3v_4, ..., v_{k-2}v_{k-1}, v_{k-1}v_k$$

Where

- $v_i \in V$ and $v_iv_{i+1} \in E$ for all i, and
- $v_1 = s, v_k = t$ and $\underbrace{v_i \neq v_j \text{ for all } i \neq j}_{\text{Without this, it is called an s,t-walk}}$

The length of a path $P = v_1v_2, v_2v_3, v_3v_4, ..., v_{k-1}v_k$ is the sum of the lengths of the edges on P

$$c(P) = \sum(c_e : e \in P)$$

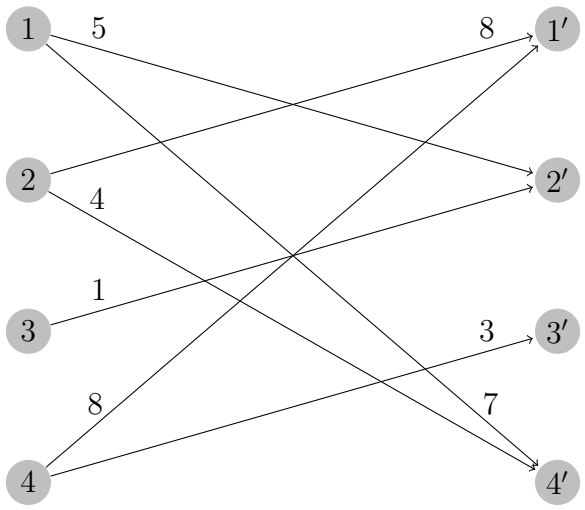
4.2 IP Models: Matchings

Example:
WaterTech has a collection of important jobs that it needs to handle urgently.
It also has 4 employees that need to handle these jobs.
Employees have different skill-sets and may take different amount of times to execute a job, note that some workers are not able to handle certain jobs!

Goal: Assign each worker to exactly one task so that the total execution time is smallest!
We will rephrase this in the language of graphs.

Employees	Jobs			
	1'	2'	3'	4'
1	-	5	-	7
2	8	-	-	4
3	-	1	-	-
4	8	-	3	-

Create a graph with one vertex for each employee and job.
Add an edge ij with cost c_{ij} for $i \in E$ and $j \in J$ if employee i can handle job j in time c_{ij}



Matching

- A collection $M \subseteq E$ is matching if no two edges $ij, i'j' \in M (ij \neq i'j')$ share an endpoint.
- i.e. $\{ij\} \cap \{i'j'\} = 0$

For example:

- $M = \{14', 21', 32', 43'\}$ is a matching
- $M = \{14', 32', 41', 43'\}$ is NOT a matching

The cost of a matching M is the sum of costs of its edges:

$$c(M) = \sum (c_e : e \in M)$$

A matching M is perfect if every vertex v in the graph is incident to an edge in M
Note: a perfect matching correspond to feasible assignments of workers to jobs!

More notations:
Use $\delta(v)$ to denote the set of edges incident to v, i.e.:

$$\delta(v) = \{e \in E : e = vu \text{ for some } u \in V\}$$

This definition is improved later!
For example:

- $\delta(2) = \{21', 24'\}$
- $\delta(3') = \{43'\}$

So another definition of a perfect matching is:

- Given $G = (V, E), M \subseteq E$ is a perfect matching iff $M \cap \delta(v)$ contains a single edge for all $v \in V$

So the IP will have a binary variable x_e for every edge $e \in E$, the idea is:

- $x_e = 1 \longleftrightarrow e \in M$

So the constraints for perfect matching is:
For all $v \in V$, we need

$$\sum(x_e : e \in \delta(v)) = 1$$

The objective function would be:

$$\sum(c_e x_e : e \in E)$$

Complete IP for any perfect matching problem:

$$\begin{array}{rl} \min & \sum(c_e x_e : e \in E) \\ \text{s.t} & \sum(x_e : e \in \delta(v)) = 1 (v \in V) \\ & x \geq 0, \text{ x is integer} \end{array}$$

For the example question, we have:

$$\begin{array}{rl} \min & (5, 1, 3, 4)x) \\ \text{s.t} & \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{pmatrix} \begin{array}{cccc} 12 & 13 & 14 & 23 \end{array} \\ \begin{array}{cccc} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{array} \end{pmatrix} x = \mathbf{1} \\ & x \geq 0, \text{ x is integer} \end{array}$$

4.3 Shortest Path Problem

Input:

- Graph $G = (V, E)$
- Non-negative edge lengts c_e for all $e \in E$
- Vertices $s, t \in V$

Goal: Compute an s, t-path of the smallest total length

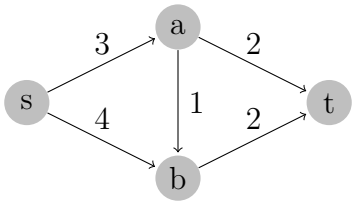
The shortest path problem is:

- Given: Graph $G = (V, E)$, lengths $c_e \geq 0$ for all $e \in E$, and $s, t \in V$, compute an s, t-path of smallest total length

Useful Observation:

- Let $C \in E$ be a set of edges whose removal disconnects s and t

For example:



Let $C = \{sb, ab, at\}$, notice that removing all edges in C eliminates all paths from s to t
Therefore, every s,t-path must have at least one edge in C

A more precise definition of notation δ:
For $S \in V$, we let $\delta(S)$ be the set of edges with exactly one endpoint in S

$$\delta(S) = \{uv \in E : u \in S, v \notin S\}$$

Examples:

1. $S = \{s\} \to \{sa, ab\}$
2. $S = \{s, a\} \to \{ab, at, sb\}$
3. $S = \{a, b\} \to \{sa, sb, at, bt\}$

Definition of s,t-cut:

- $\delta(S)$ is an s,t-cut if $s \in S$ and $t \notin S$

Using the 3 $\delta(S)$ examples above, 1 and 2 are s,t-cuts, 3 is not

Remark:

1. if P is an s,t-path and $\delta(S)$ is an s,t-cut, then P must have an edge from $\delta(S)$
2. if $S \subseteq E$ contains at least one edge from every s,t-cut, then S contains an s,t-path

Prove #2 by contradiction:

- suppose S has an edge from every s,t-cut, but S has no s,t-path
- Let R be the set of vertices reachable from s in S: $R = \{u \in V : S \text{ has an } s,u \text{ path}\}$
- $\delta(R)$ is an s,t-cut since $s \in R$ and $t \notin R$
- Note: there cannot be an edge $uv \in S$ with $u \in R$ and $v \notin R$. Otherwise v should have been in R!
- So $\delta(R) \cap S = \emptyset$
- Contradiction!

Generic IP for Shortest Path problem:
Variables:
We have one binary variable x_e for each edge $e \in E$. We want:

$$x_e = \begin{cases} 1 & : e \in P \\ 0 & : \text{otherwise} \end{cases}$$

Constraints:
We have one constraint for each s,t-cut $\delta(U)$, forcing P to have an edge from $\delta(S)$

$$\sum x_e : e \in \delta(U)) \geq 1 \qquad \text{for all } s,t\text{-cuts } \delta(U) \tag{1}$$

Objective Function:

$$\sum (c_e x_e : e \in E)$$

Complete Model:

$$\begin{array}{ll} \min & \sum (c_e x_e : e \in E) \\ \text{s.t.} & \sum (x_e : e \in \delta(U)) \geq 1 (U \subseteq V, s \in U, t \notin U) \\ & x_e \geq 0,\ x_e \text{ integer} \end{array}$$

Suppose $c_e > 0$ for all $e \in E$, then in an optimal solution, $x_e \leq 1$ for all $e \in E$

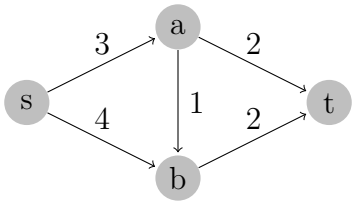
- Suppose $x_e \geq 1$
- Then let $x_e = 1$. This is cheaper and maintains feasibility!

For binary solution x, define

$$S_x = \{e \in E : x_e = 1\}$$

Note: If x is a feasible for an IP, then S_x has at least one edge from every s,t-cut and S_x has a s,t-path, but S_x may contain more than just an s,t-path! Consider this diagram

again:



Let $x_e = 1$ for $e \in \{sa, ab, at\}$, and $x_e = 0$ otherwise.
So $S_x = \{sa, ab, at\}$
x cannot be optimal for the IP because we can reduce S_x and get a better solution!
i.e. let $x_a b = 0$ and the solution is more efficient

So if x is an optimal solution for the IP and $c_e \geq 0$ for all $e \in E$ then S_x contains the edges of a shortest s, t-path

5 Nonlinear Programs (NLP)

A nonlinear program (NLP) is of the form:

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & g_1(x) \leq 0 \\ & g_2(x) \leq 0 \\ & \dots \\ & g_m(x) \leq 0 \end{array}$$

Where:

- $x \in \mathbb{R}$
- $f : \mathbb{R}^n \to \mathbb{R}$
- $g_i : \mathbb{R}^n \to \mathbb{R}$

Note: Linear programs (LPs) are NLPs!

5.1 NLP Models: Finding Close Points in an LP

Problem:
We are given an LP (P), and an infeasible point $\bar{x}$
Goal:
Find a point $x \in P$ that is as close as possible to $\bar{x}$

i.e. find a point $x \in P$ that minimizes the Euclidean distance to $\bar{x}$

$$||x - \bar{x}||_2 = \sqrt{\sum_{i=1}^{n} (x_i - \bar{x}_i)^2}$$

Remark: $||p||_2$ is called the L^2-norm of p

Generic model:

Given LP:		Solve:	
min	$c^T x$	min	$||x - \bar{x}||_2$
s.t.	$x \in P$	s.t.	$x \in P$ where $P = \{x : Ax \leq b\}$

5.2 NLP Models: Binary IPs

Suppose we are given a binary IP (i.e. an IP with all variables are binary)
Recall that Binary IPs are generally hard to solve!
We need to rewrite binary IPs as NLPs

Generic model:

Given IP:			Rewrite:			
max	$c^T x$		max	$c^T x$		
s.t.	$Ax \leq b$		s.t.	$Ax \leq b$		
	$x \geq 0$			$x \geq 0$		
	$x_j \in \{0,1\}$	$(j \in \{1, /dots, n\})$		$x_j(1-x_j) = 0$	$(j \in [n])$	$(*)$

Correctness:
For $j \in [n]$, $(*)$ holds iff $x_j = 0$ or $x_j = 1$

Question:
Can you change the NLP to express teh fact that x_j is any non-negative integer instead of binary?

$$\begin{array}{ll} \max & c^T x \\ \text{s.t.} & Ax \leq b \\ & x \geq 0 \\ & sin(\pi x_j) = 0 \qquad (j \in [n]) \qquad (**) \end{array}$$

Correctness:
Note that $sin(\pi x_j) = 0$ only if x_j is an integer. Combining with non negative constraint it limits x_j to be any non-negative integer.

5.3 Fermat's last Theorem

Conjecture:
There are no integers $x, y, z \geq 1$ and $n \geq 3$ such that

$$x^n + y^n = z^n$$

The proof is 150 pages long!

NLP for Fermat's Last Theorem

$$\begin{array}{ll} \min & (x_1{}^{x_4} + x_2{}^{x_4} - x_3{}^{x_4})^2 + (sin\ \pi x_1)^2 + (sin\ \pi x_2)^2 + (sin\ \pi x_3)^2 + (sin\ \pi x_4)^2 \\ \text{s.t.} & x_i \geq 1 \qquad (i = 1 \ldots 3) \\ & x_4 \geq 3 \end{array}$$

- This NLP is trivially feasible, and the value of any feasible solution is non-negative as its objective is the sum of squares, which means the min possible value is 0
- In fact, the value of a solution (x_1, x_2, x_3, x_4) is 0 iff
  - ${x_1}^{x_4} + {x_2}^{x_4} = {x_3}^{x_4}$, and
  - $(sin\ \pi x_i)^2 = 0$, for all $(i = 1 \ldots 3)$

Notes:

- Fermat's Last Theorem is true iff the NLP has optimal value greater than 0
- It is well known that there is an infinite sequence of feasible solutions whose objective value converges to 0!
- To prove Fermat's Last Theorem we must show that the value 0 can not be attained!