27, August 2024

CSE, IIT KGP

Linear Programming

Computing Lab (CS 69201)
Department of Computer Science and Engineering
IIT Kharagpur

Basic Info

- Linear programming (LP) is a mathematical optimization technique used to find the best outcome (such as maximum profit or minimum cost) in a model defined by linear relationships.

- Sounds boring and complex ??

Let's be practical

- Maximize $3x + 4y$

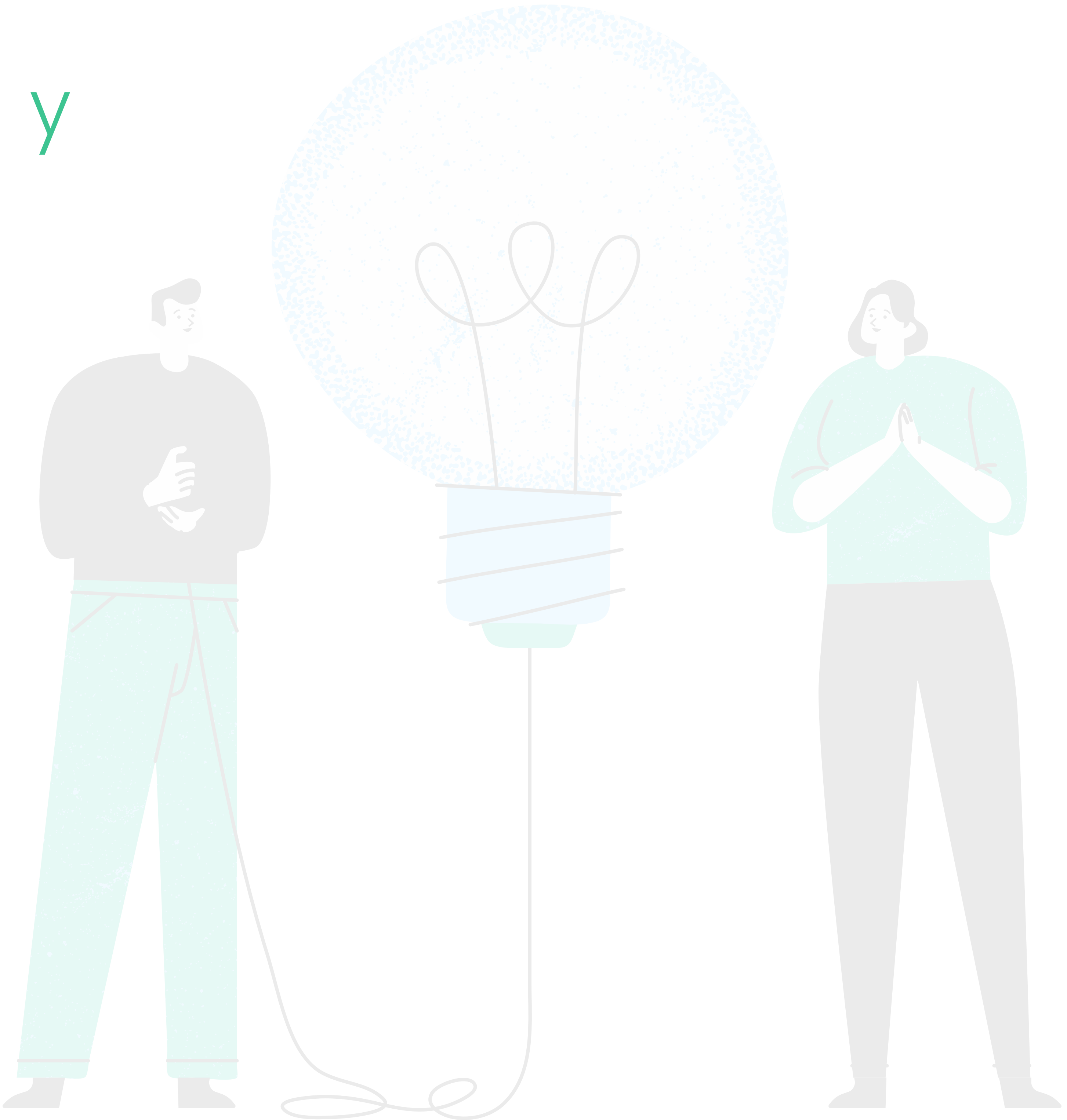
Objective function

Decision variables - x and y

Subject to the conditions

- $x+2y <= 14$
- $3x-y >= 0$
- $x-y <= 2$

Constraints



How do we solve them

- Some complex maths ?

Let's get started !!!

And we will do in python !!!

Google OR-Tools

1. Installing the package

- You will need to use the pywraplp module from the ortools package. For that install the ortools package.

- The ortools is actually written in C++.We will use the python wrapper over it.

```
pip install ortools
```

2. Import the required libraries

- You will need to import the pywraplp module from the ortools.linear_solver package.

```
from ortools.linear_solver import pywraplp
```

3. Declare the solver

- Create a solver variable that will contain all the necessary items to solve the problem.

- You can do this by calling the CreateSolver() method of the pywraplp solver class and passing in the name of the solver you want to use. For linear programming problems, you can use the GLOP solver.

```
solver = pywraplp.Solver.CreateSolver("GLOP")
```

4. Create the variables

- Create the variable by calling the NumVar()/IntVar() method of the solver object and passing in the lower and upper bounds for the variable, as well as a name for the variable.

```
x = solver.NumVar(0, solver.infinity(), "x")
y = solver.NumVar(0, solver.infinity(), "y")
```

Lower bound

Upper bound

5. Define the constraints

- Now to define the constraints ,we will call the Add() method of the solver object and passing in a linear expression that represents the constraint.

```
# Constraint 0: x + 2y <= 14.
solver.Add(x + 2 * y <= 14.0)

# Constraint 1: 3x - y >= 0.
solver.Add(3 * x - y >= 0.0)

# Constraint 2: x - y <= 2.
solver.Add(x - y <= 2.0)
```

5. Define the constraints (Alternative way)

- Alternately, You can do this by calling the Constraint() and SetCoefficient() method of the solver object that represents the constraint.

```
# Create a linear constraint 0<=x+y<=2
ct = solver.Constraint(0,2,"ct")
# Set coefficients for the variables in the constraint
constraint.SetCoefficient(x, 1)
constraint.SetCoefficient(y, 2)
```

6. Define the objective function

- Now for optimization , call either the Maximize() or Minimize() method of the solver object, depending on whether you want to maximize or minimize the objective function, and passing in a linear expression that represents the objective function.

```
# Objective function: 3x + 4y.
solver.Maximize(3 * x + 4 * y)
```

6. Define the objective function (Alternative way.)

- Alternately, you can represent the objective function using Objective() and SetCoefficient() methods of the solver object.

```
# Define the objective function
objective = solver.Objective()

# Set coefficients for the objective function
objective.SetCoefficient(x, 3)  # Coefficient of p
objective.SetCoefficient(y, 4)  # Coefficient of q

# Set the objective to maximize
objective.SetMaximization()
```

7. Invoke the solver

- After defining all of the necessary components of your problem, you can solve it by calling the solve() method of the solver object.

```
status = solver.Solve()
```

8. Display the solution

- Finally, you can display the solution to your problem by accessing the values of your variables using their solution_value() attribute for the variables.

```
print(f"Objective value = {solver.Objective().Value():0.1f}")
print(f"x = {x.solution_value():0.1f}")
print(f"y = {y.solution_value():0.1f}")
```

Let’s see a more **practical** situation !!!

Food items Available

Food Item	Cost (Per unit in $)	Calories (gm)	Protein (gm)	Carbohydrates (gm)	Fat (gm)
Chicken	5	250	30	0	10
Rice	2	200	5	45	1
Broccoli	1	50	4	10	0.5
Salmon	8	300	25	0	20
Quinoa	4	220	8	39	4

You are a nutritionist tasked with creating a healthy meal plan for a client while minimising costs. You have several food items available, each with a cost, calorie, protein, carbohydrate, and fat content. The goal is to determine the optimal quantities of each food item to include in the meal plan to meet the client's nutritional needs while minimising the total cost

Nutritional Requirements:

- Total Calories: At least 1500 calories
- Total Protein: At least 70 grams
- Total Carbohydrates: At least 150 grams
- Total Fat: At least 40 grams

How can you determine the optimal quantities of each food item to include in the meal plan to minimise the total cost ? And what are the optimal quantities for each food item to meet the nutritional requirements?

Problem Formulation

Let:

- $x1$: Quantity of Chicken
- $x2$: Quantity of Rice
- $x3$: Quantity of Broccoli
- $x4$: Quantity of Salmon
- $x5$: Quantity of Quinoa

Objective Function

Minimize the total cost: $5x1+2x2+1x3+8x4+4x5$

Constraints

- Caloric Requirement:
  $250x_1+200x_2+50x_3+300x_4+220x_5 \geq 1500$
- Protein Requirement: $30x_1+5x_2+4x_3+25x_4+8x_5 \geq 70$
- Carbohydrate Requirement: $45x_2+10x_3+39x_5 \geq 150$
- Fat Requirement: $10x_1+1x_2+0.5x_3+20x_4+4x_5 \geq 40$
- Non-negativity: $x_1, x_2, x_3, x_4, x_5 \geq 0$

Now write down this program!!!

& thats

~ Q1 of today’s assignment

LP Relaxation

There are four possible projects, which each run for 3 years and have the following characteristics.

		Capital Requirements for Each Year		
Project	Return	1st year	2nd year	3rd year
1	0.2	0.5	0.3	0.2
2	0.3	1.0	0.8	0.2
3	0.5	1.5	1.5	0.3
4	0.1	0.1	0.4	0.1
Available Capital		3.1	2.5	0.4

Which projects would you choose in order to maximise the total return?

Branch & Bound with LP relaxation

Consider the following problem:

Objective to maximize: $0.2x_1 + 0.3x_2 + 0.5x_3 + 0.1x_4$.

subject to

$0.5x_1 + 1.0x_2 + 1.5x_3 + 0.1x_4 <= 3.1$ (i)
$0.3x_1 + 0.8x_2 + 1.5x_3 + 0.4x_4 <= 2.5$ (ii)
$0.2x_1 + 0.2x_2 + 0.3x_3 + 0.1x_4 <= 0.4$ (iii)
$x_j = 0$ or 1 $j=1,...,4$

The solution given by the package is : $x_2=0.5$, $x_3=1$, $x_1=x_4=0$

Branch & Bound with LP relaxation

The package returns fractional value for the answer.

How can we rid ourselves of this troublesome fractional value?

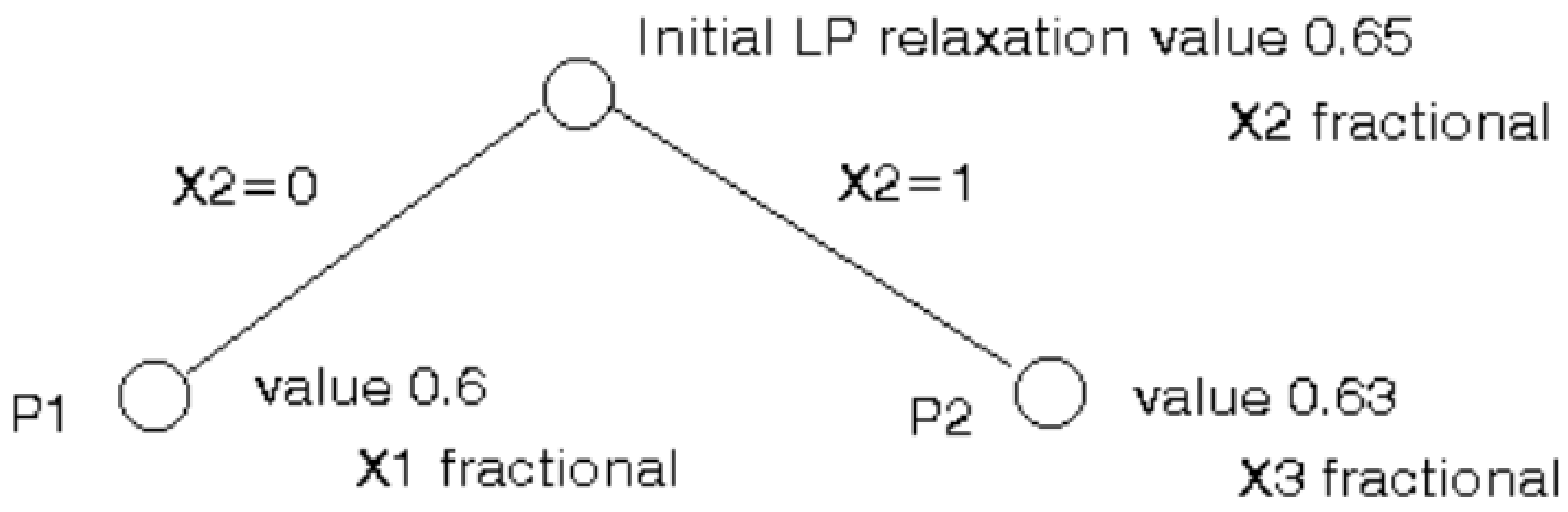
To remove this troublesome fractional value we can generate two new problems:

original LP relaxation plus x2=0
original LP relaxation plus x2=1

Branch & Bound with LP relaxation

We now have two new LP relaxations to solve. If we do this we get:

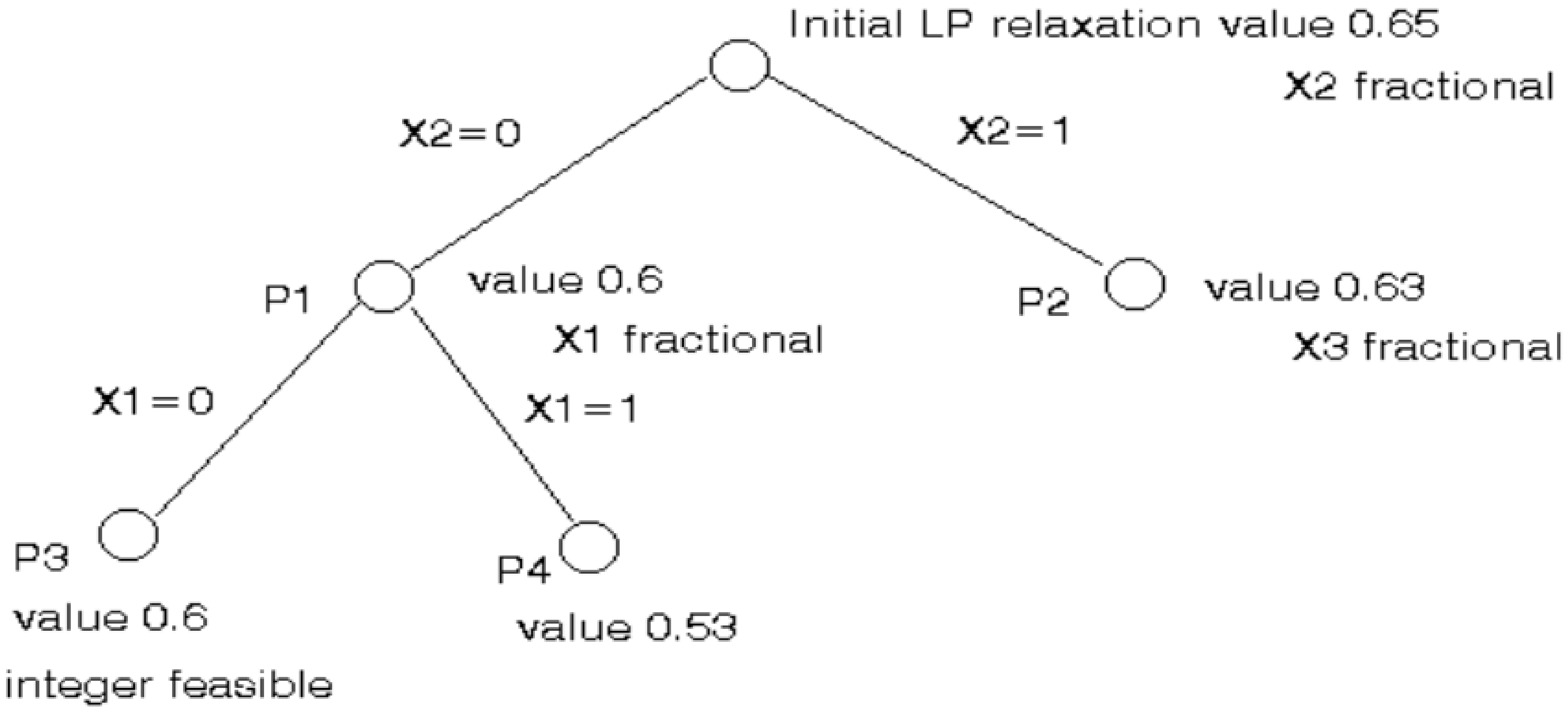
- P1 - original LP relaxation plus x_2=0, solution x_1=0.5, x_3=1, x_2=x_4=0 of value 0.6
- P2 - original LP relaxation plus x_2=1, solution x_2=1, x_3=0.67, x_1=x_4=0 of value 0.63



Branch & Bound with LP relaxation

In the solution of P1, x1 is fractional hence, we further branch on x1 from P1.

In the solution of P2, x3 is fractional hence, we further branch on x3 from P2.



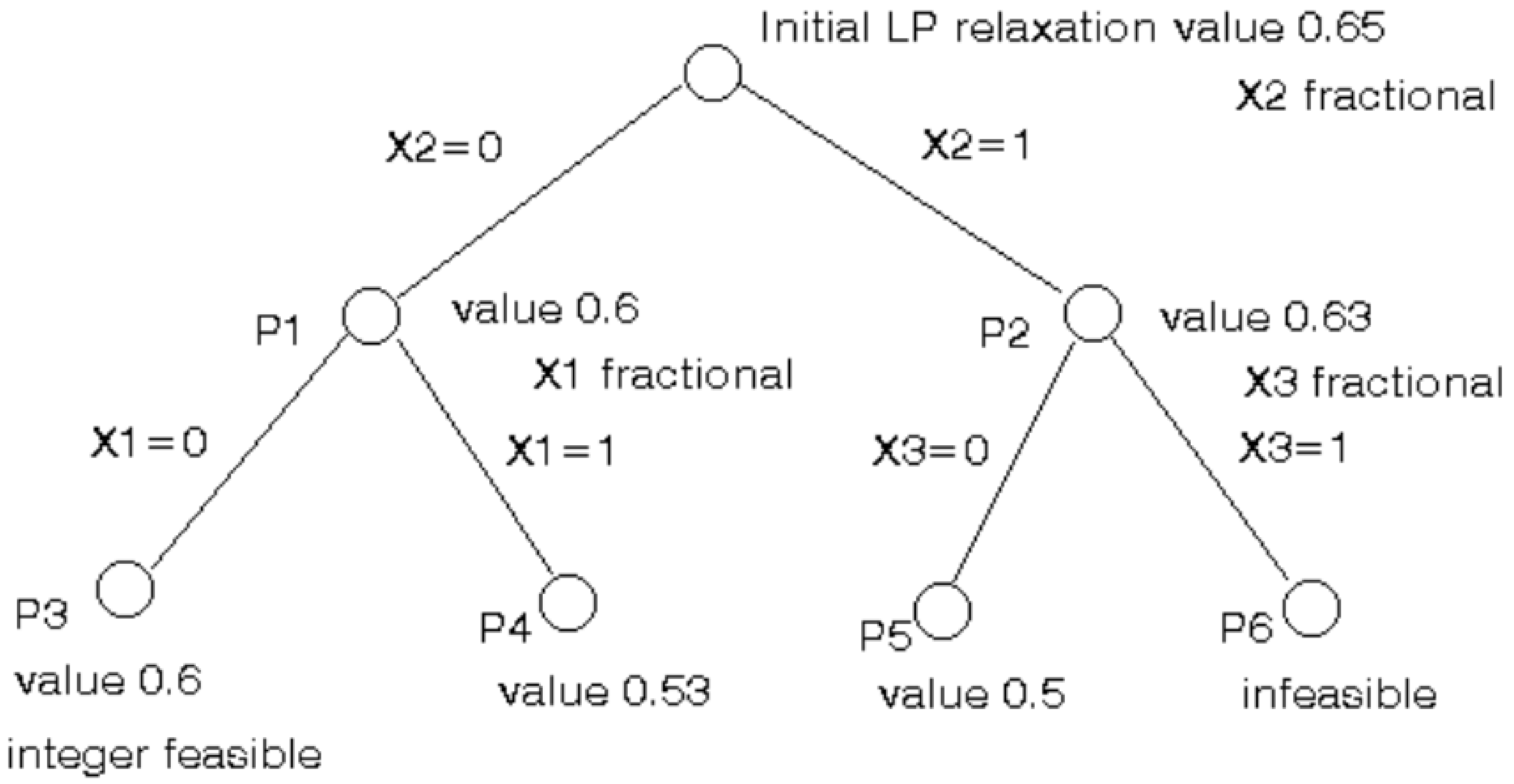
Branch & Bound with LP relaxation

Choosing problem P1 we branch on x_1 to get our list of LP relaxations as:

- P3 - original LP relaxation plus $x_2=0$ (P1) plus $x_1=0$, solution $x_3=x_4=1$, $x_1=x_2=0$ of value 0.6
- P4 - original LP relaxation plus $x_2=0$ (P1) plus $x_1=1$, solution $x_1=1$, $x_3=0.67$, $x_2=x_4=0$ of value 0.53
- P2 - original LP relaxation plus $x_2=1$, solution $x_2=1$, $x_3=0.67$, $x_1=x_4=0$ of value 0.63

Branch & Bound with LP relaxation

Consider P4, it has value 0.53 and has a fractional variable (x_3). However if we were to branch on x_3 any objective function solution values we get after branching can never be better (higher) than 0.53



Branch & Bound with LP relaxation

Branching on x_3 we get

- P5 - original LP relaxation plus x_2=1 (P2) plus x_3=0, solution x_1=x_2=1, x_3=x_4=0 of value 0.5.
- P6 - original LP relaxation plus x_2=1 (P2) plus x_3=1, problem infeasible as constraint (iii) fails.

Neither of P5 or P6 lead to further branching so we are done, we have discovered the optimal integer solution of value 0.6 corresponding to x_3=x_4=1, x_1=x_2=0.

Branch & Bound with LP relaxation

To use this efficiently, we can use the following

x= IntVar(initialvalue, finalvalue, name)

- NumVar( ) - gives a continuous variable. It can take any real number within a specified range.

- IntVar( ) - gives an Integer variable. It Can only take integer values within a specified range.

So what to do now ?

Some Inclass assignments on linear programming and may be some more to do in the home (?)

Method: Use OR tools

No of Question 4

Duration 2 hours

Location CSE LAB (2nd Floor)

Thank you
for listening!