MSBA 403:

Optimization

Lecture 2

November 13, 2020

UCLA Anderson School of Management

Lecture 1 Recap

- Structure of a mathematical program
  - Decision variables
  - Constraints
  - Objective (minimize or maximize)

- Examples of linear programs (LPs)
  - Bag production
  - Cargo revenue management
  - Product delivery

- Geometry of LPs and simplex algorithm
  - Constraints define a “feasible region” for the LP
  - One of the corner points (extreme points) of the feasible region is always optimal

Sensitivity analysis

- The purpose of performing a *sensitivity analysis* is to understand how the optimal value of an optimization problem varies as one of the parameters changes

- Doing this gives us an idea of the value of a particular resource

- In linear programs, the **shadow price** of a constraint describes the change in the optimal value from a small increase in the right hand side of the constraint

- **Let's see how to perform sensitivity analyses in Jupyter**

Absolute value functions in LPs

- What if the objective function contains absolute values? E.g.,

$$\min \ |f(x)|$$

- Note that $|f(x)| \leq z$ if and only if $f(x) \leq z$ and $f(x) \geq -z$

- Then we have

(Non-linear) $\min |f(x)| \iff$ $\begin{aligned} &\min z \\ &\text{s.t. } z \geq f(x) \\ &\quad\;\; z \geq -f(x) \end{aligned}$ **(Linear!)**

Example: Inventory management

Suppose you are managing the inventory of a warehouse over periods $i = 1, 2, \ldots, N$, and you are required to meet demand d_i in each period i. You need to decide how much product to order in each period. There are two costs involved:

- For every unit of product held in inventory from the end of period i to the beginning of period $i + 1$, you incur a holding cost of c_1.

- if you change the quantity of product ordered in i to a different amount in $i + 1$, you incur a cost that is equal to c_2 multiplied by the absolute value of the change. For example, if your order quantity from one period to the next increases or decreases by 5 units, the associated order change cost is $5{*}c_2$.

Assume you start with zero inventory. Formulate a linear programming model for this problem.

Example: Inventory management

A cost minimizing solution is given by the LP:

$$
\begin{aligned}
\min \quad & c_1 \sum_{i=1}^{N} s_i + c_2 \sum_{i=1}^{N} y_i && \leftarrow \text{Minimize total cost} \\
\text{s.t.} \quad & s_1 = x_1 - d_1 && \leftarrow \text{Inventory balance constraint for period 1} \\
& s_i = s_{i-1} + x_i - d_i, \quad i = 2, \ldots, N, && \leftarrow \text{Inventory balance constraint for periods 2...N} \\
& \left.\begin{aligned} & y_i \geq x_i - x_{i+1}, \quad i = 1, \ldots, N, \\ & y_i \geq x_{i+1} - x_i, \quad i = 1, \ldots, N, \end{aligned}\right\} && \text{Absolute deviation from previous order quantity} \\
& x_i, y_i, s_i \geq 0, \quad i = 1, \ldots, N.
\end{aligned}
$$

Integer programming

- In linear programs, all decision variables are continuous

- We might also have an optimization problem where decisions are inherently discrete, e.g. “yes or no” type decisions

- This requires the use of **binary variables**

$$x_i = \begin{cases} 0, & \text{if no} \\ 1, & \text{if yes} \end{cases}$$

- Binary variables allow us to model an broad range of decisions
  - Should we build a warehouse at location i?
  - Should we buy ads in market i?
  - Should we invest in project i?
  - Should we assign surgery i to operating room j at time k?

- Optimization models with binary variables are called **integer programs**
- More specifically, when the objective an all constraints are linear, the model is an **integer linear program**

Example: Fleet assignment at Delta Air Lines

- In 1994, Delta Air Lines flew 2,500 domestic flight legs each day, using 450 different aircraft from 10 different fleet types

- Every empty seat on an airline is lost revenue -- the goal is therefore to determine a schedule that captures as much business as possible

- A major component of an airlines schedule is the **fleet assignment problem,** which is to determine which aircraft should be assigned to each flight leg

- Delta developed a mixed-integer optimization model to solve the fleet assignment problem, which was estimated to save $100 million per year

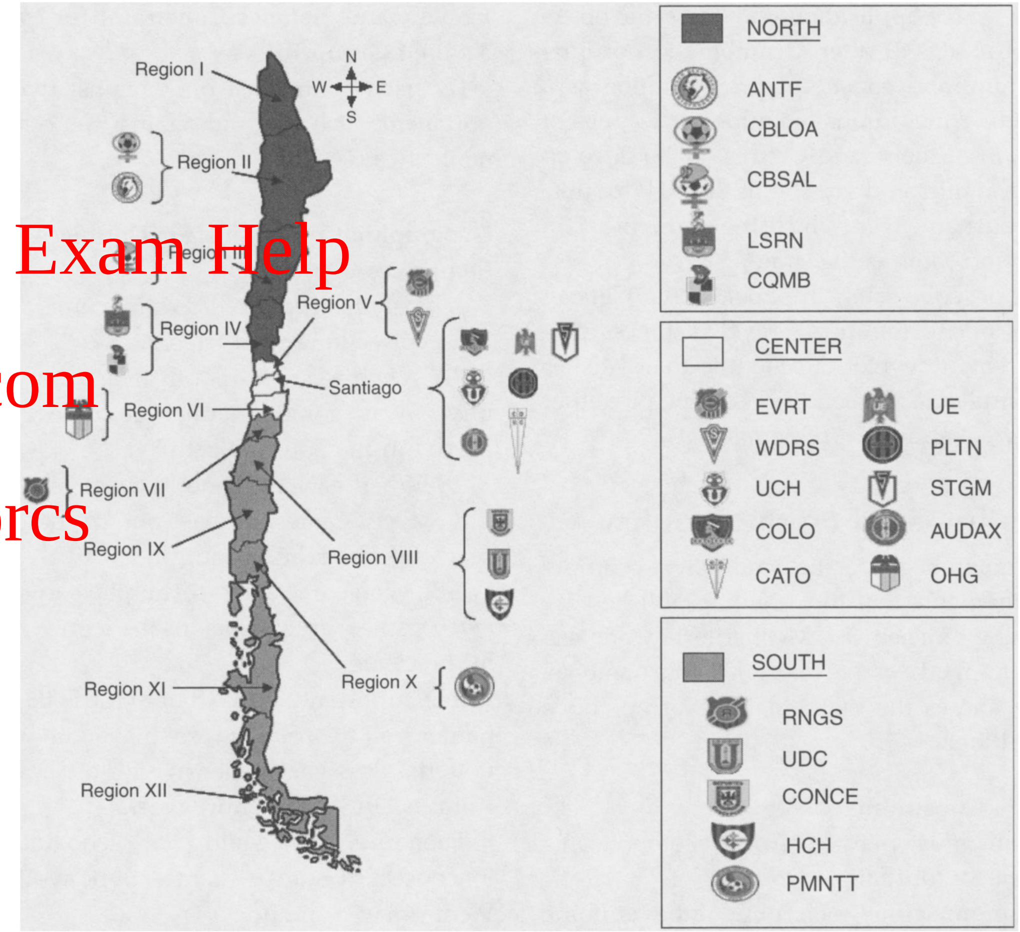
Example: Chilean soccer league

- Sports scheduling: Which teams should play each other, and when?

- Many factors to consider, including
  - Balance (equal number of “home” and “away” games)
  - Fairness (unfair for some teams to travel much larger distances than other teams)
  - Economic (Ticket revenue might be higher on Friday evening vs Tuesday evening)
  - Appeal (scheduling two strong teams late in the season generates more excitement)

- Since 2005, the Chilean soccer league has been constructing the schedule using integer linear programming

- Ticket revenue increased from 6 million to 12 million pesos from 2004 to 2006, due to improved attendance



Example: Virginia Court of Appeals

- Court systems need to assign judges to sessions

- Virginia's Court of Appeals constructed judge schedules for the year manually, taking into account
  - Timing and location of sessions
  - Availability of judges on certain days
  - Balancing judge workload
  - Each judge much sit with every other judge at least once during the year
  - Etc.

- Construction of schedule by hand took a single employee 150 hours over a few months
  - "Small" requested changes can ripple through the schedule

- Starting in 2011, the Virginia Court of Appeals an integer linear programming approach for judge assignment, which could be solved in 1 day, saving an employee months of work

- Using optimization also increases fairness of the judicial system by removing human discretion

Example: Real estate development

- An LA-based real estate development firm is considering five different projects, one in each of the following locations: Santa Monica, Westwood, Venice, West Hollywood, and Marina Del Rey.

- The following table describes the estimated profit and the investment required in each project (in millions):

	Santa Monica	Westwood	Venice	West Hollywood	Marina Del Rey
Estimated Profit	1	1.8	1.6	0.8	1.4
Investment Required	6	12	10	4	8

- The firm has raised \$20 million of investment capital for these projects, and is now deciding which projects to invest in.

- Formulate an integer program to maximize profit without exceed the \$20 million budget.

Example: Real estate development

- An optimal development plan is given by a solution to the following integer program:

$$
\begin{aligned}
\max \quad & x_1 + 1.8x_2 + 1.6x_3 + 0.8x_4 + 1.4x_5 && \longleftarrow \text{Profit} \\
\text{s.t.} \quad & 6x_1 + 12x_2 + 10x_3 + 4x_4 + 8x_5 \le 20 && \longleftarrow \text{Budget constraint} \\
& x_i \in \{0, 1\}, \quad i = 1, \ldots, 5. && \longleftarrow \text{Force all variables to be 0 or 1}
\end{aligned}
$$

Solving integer programs

- How do we solve integer programs? What if we brute force searched?
- Consider the real estate development problem: One approach is to enumerate all solutions, eliminating those that violate the budget constraint, and then compare the rest on profit. Is this a viable approach?

Suppose we had a super computer that could evaluate 1 million solutions per second.

# of projects	CPU time
20	1 second
25	1 minute
30	20 minutes
35	10 hours
40	2 weeks
45	1 year
50	35 years
55	1000 years
60	36000 years

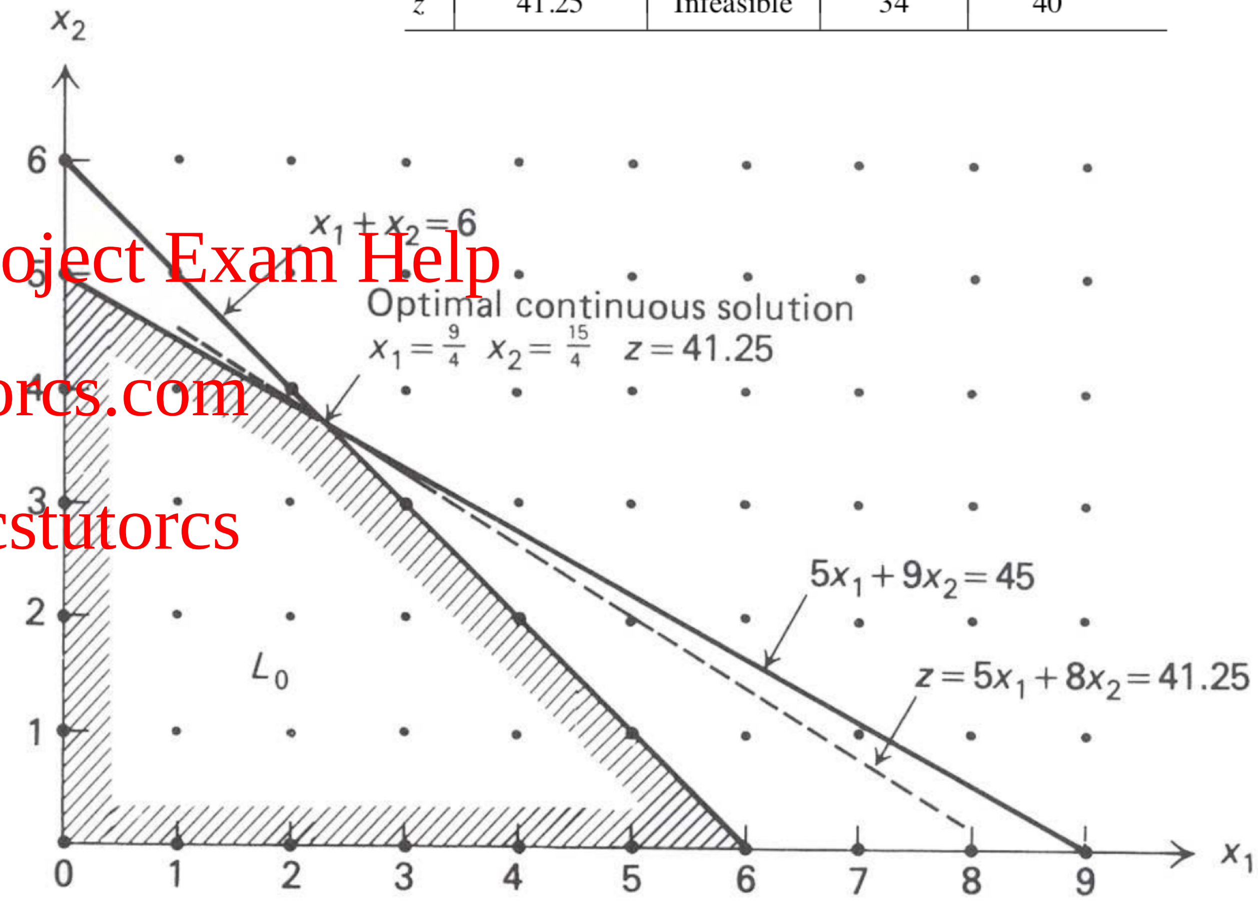
There are 2^n possible solutions! Enumeration is not feasible.
Gurobi solves the n = 60 case in 1 second. How?

Geometry of IPs

	Continuous optimum	*Round off*	*Nearest feasible point*	*Integer optimum*
x_1	$\frac{9}{4} = 2.25$	2	2	0
x_2	$\frac{15}{4} = 3.75$	4	3	5
z	41.25	Infeasible	34	40

- Consider the following simple integer program:

$$
\begin{aligned}
\max_{x_1, x_2} \quad & 5x_1 + 8x_2 \\
\text{s.t.} \quad & x_1 + x_2 \leq 6 \\
& 5x_1 + 9x_2 \leq 45 \\
& x_1, x_2 \ \text{ integer}
\end{aligned}
$$



(Figure from Bradley, Hax, Magnanti)

The branch-and-bound algorithm

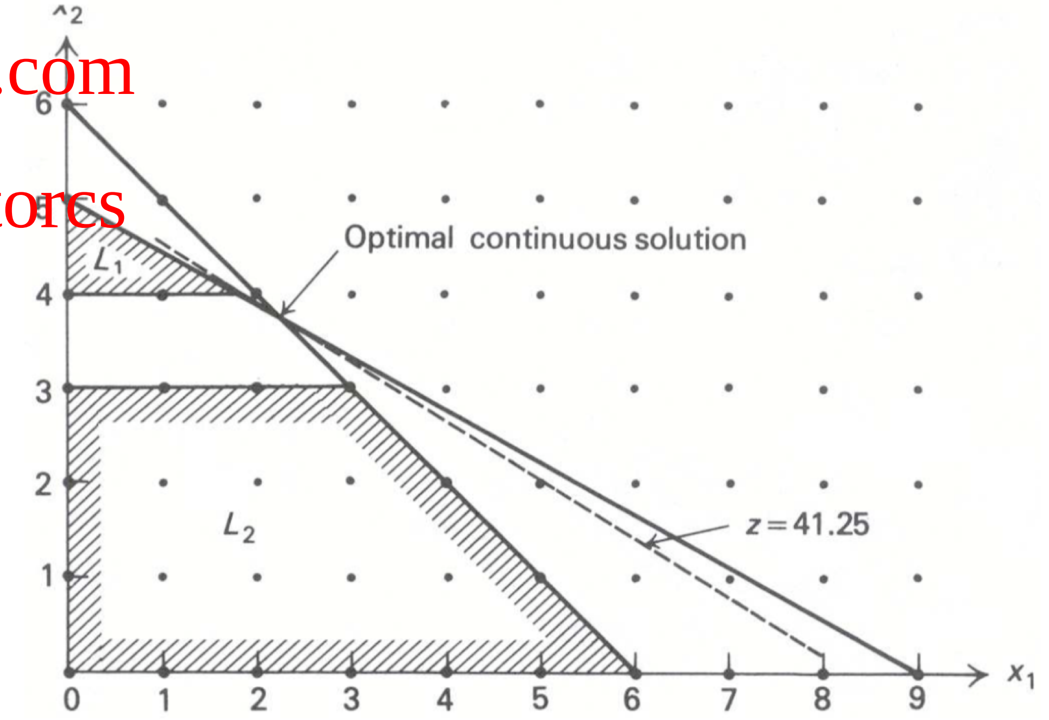
- Branch-and-bound algorithm is the engine of IP solvers (e.g. Gurobi)
- Main idea: Partition feasible region into sub-regions, and solve optimization over each sub-region ("Divide and conquer")
- The **LP relaxation** of an integer program is the LP obtained when we drop the integrality constraints from the original IP
- In a maximization (minimization) problem, the optimal value of the LP relaxation is an upper (lower) bound on the IP optimal value
  - In our example problem, the optimal solution of LP relaxation is $(x_1,x_2)=(2.25, 3.75)$, with optimal value $z = 41.25$
  - Optimal value of integer solution must therefore be ≤ 41.25

The branch-and-bound algorithm

- Because we want integer solutions, we can partition by focusing on two cases:
  $x_2 \geq 4$ and $x_2 \leq 3$. Let's call these subregions L_1 and L_2

- Let's take region L_1. Adding $x_2 \geq 4$ gives an LP relaxation solution of $(x_1,x_2)=(1.8,4)$ with optimal value $z = 41$

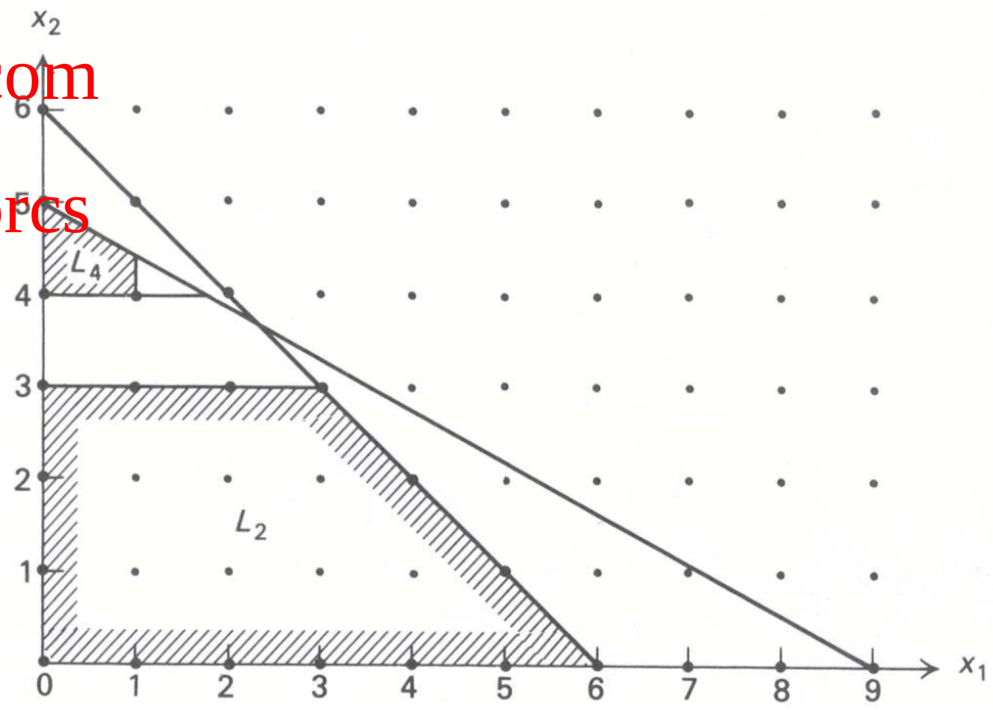
- Because x_1 is not integer, we subdivide again, producing subregion L_3 with $x_1 \geq 2$ and L_4 with $x_1 \leq 1$



The branch-and-bound algorithm

- Because we want integer solutions, we can partition by focusing on two cases: $x_2 \geq 4$ and $x_2 \leq 3$. Let's call these subregions L_1 and L_2
- Let's take region L_1. Adding $x_2 \geq 4$ gives an LP relaxation solution of $(x_1,x_2)=(1.8,4)$ with optimal value $z = 41$

- Because x_1 is not integer, we subdivide again, producing subregion L_3 with $x_1 \geq 2$ and L_4 with $x_1 \leq 1$
  - L_3 is infeasible
  - L_4 has solution $(x_1,x_2)=(1,40/9)$
  - Because x_2 is not integer in the L_4 solution, we divide again into L_5 with $x_1 \leq 4$ and L_6 with $x_2 \geq 5$



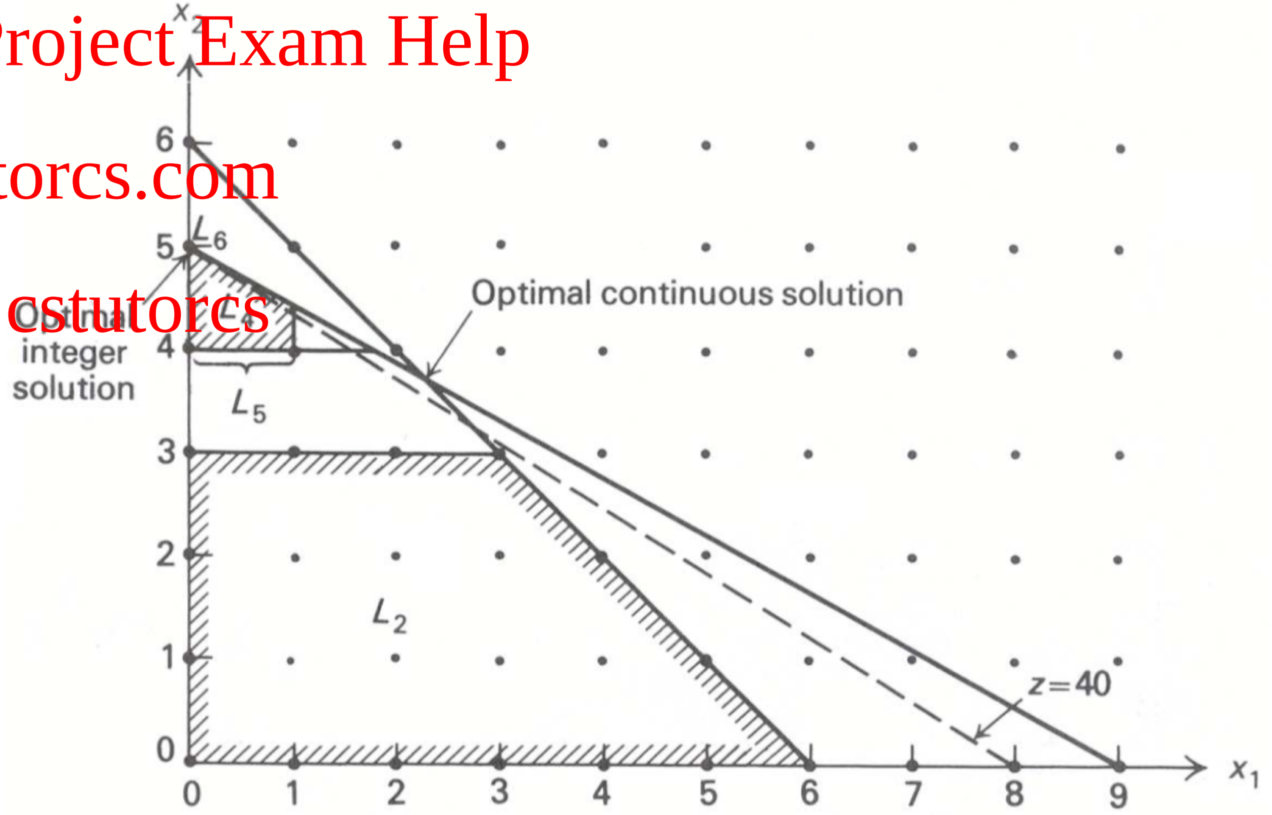
The branch-and-bound algorithm

- L_5 gives solution $(x_1,x_2)=(1,4)$, which is integer, with optimal value $z = 37$

- Because L_5 gave an integer solution, we don't need to subdivide it further

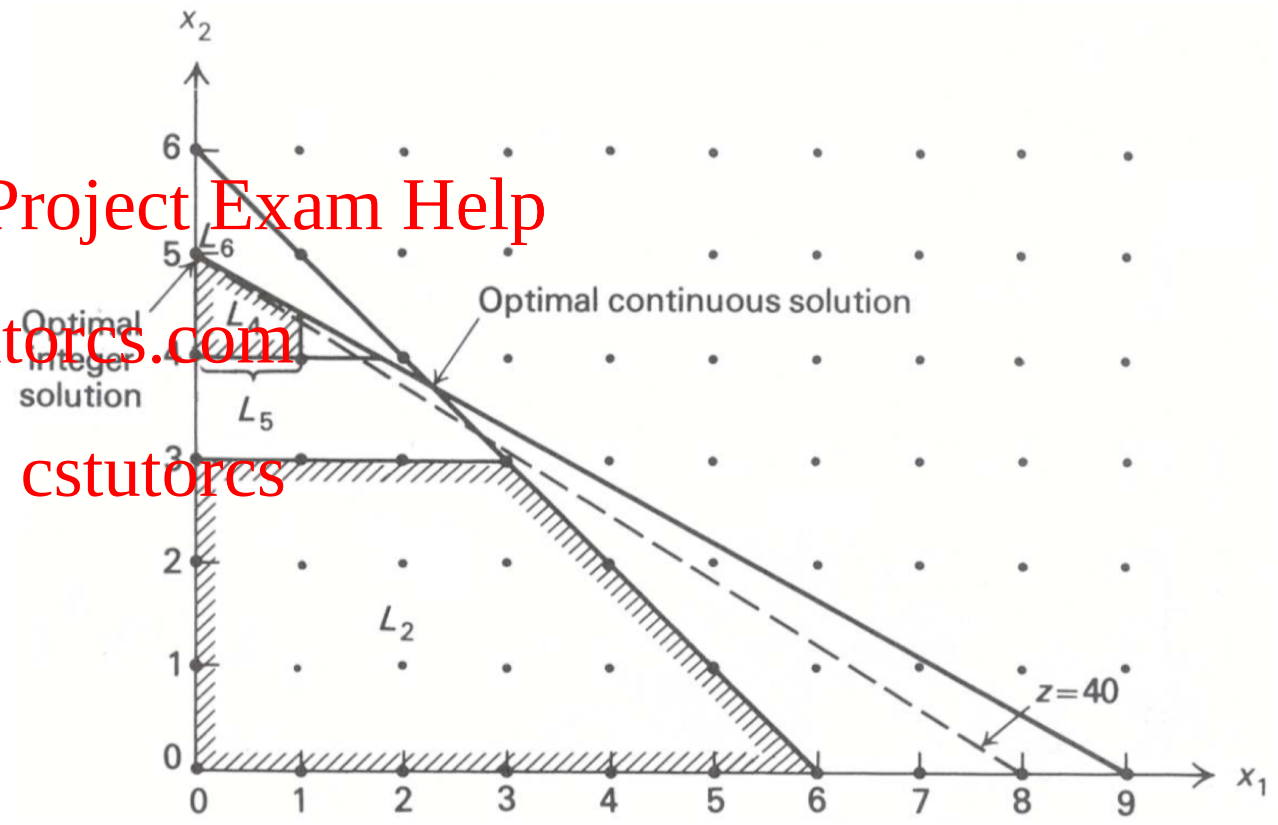
- We also know that $37 \le z^* \le 41$

- $(x_1,x_2)=(1,4)$ with $z = 37$ is the best integer solution *so far* -- we now search over L_2 and L_6 in a similar manner to see if we can do better



The branch-and-bound algorithm

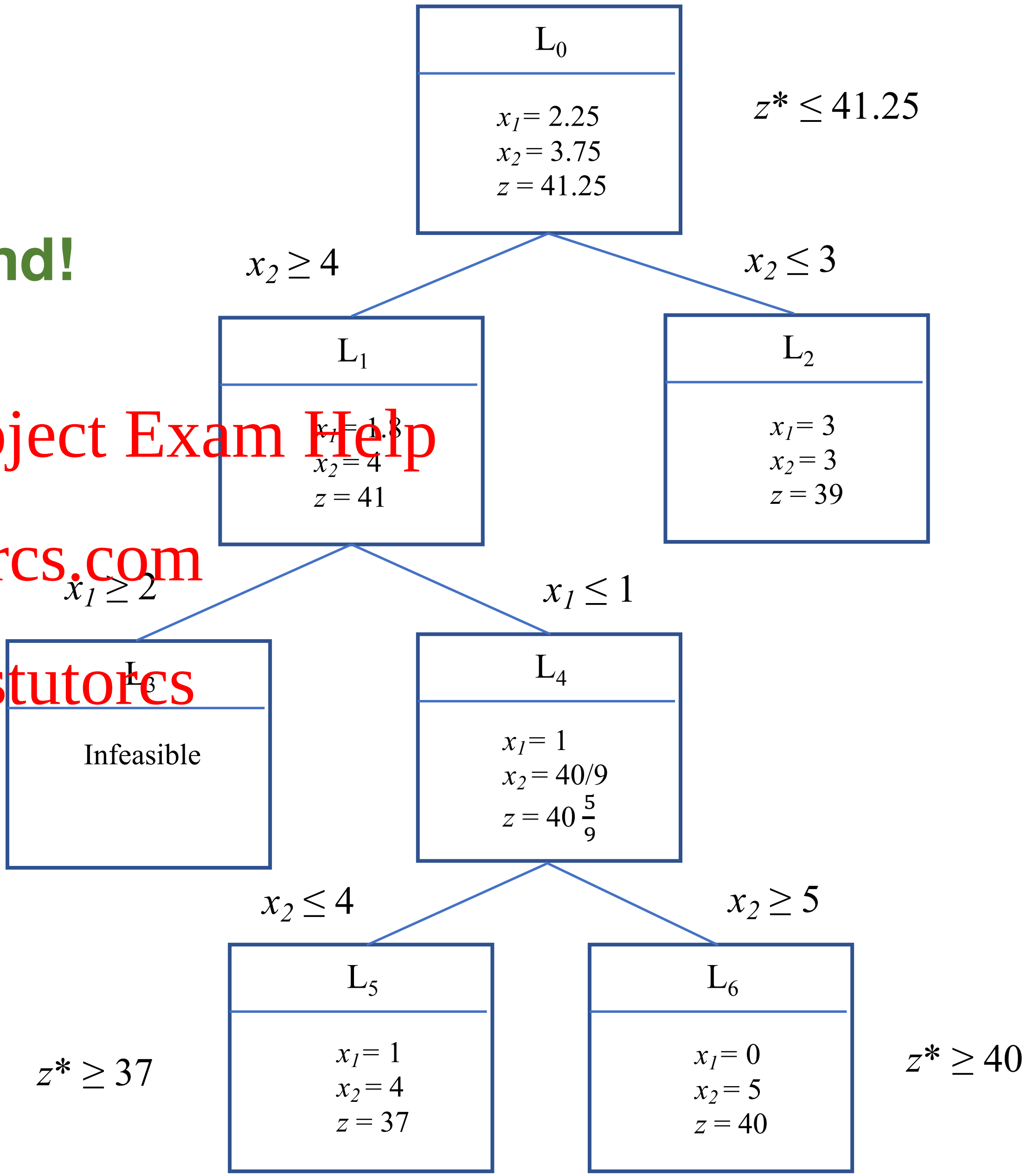
- L_6 gives solution (x_1,x_2)=(0,5) with optimal value $z = 40$
- L_2 gives solution (x_1,x_2)=(3,3) with optimal value $z = 39$
- We have searched all the subregions, so the optimal solution of the IP is the best integer solution found so far, which is (x_1,x_2)=(0,5) with $z^* = 40$



Tree representation

Optimal solution found!

LP bound: $z^* \leq 40$
Best integer solution found: $z^* \geq 40$

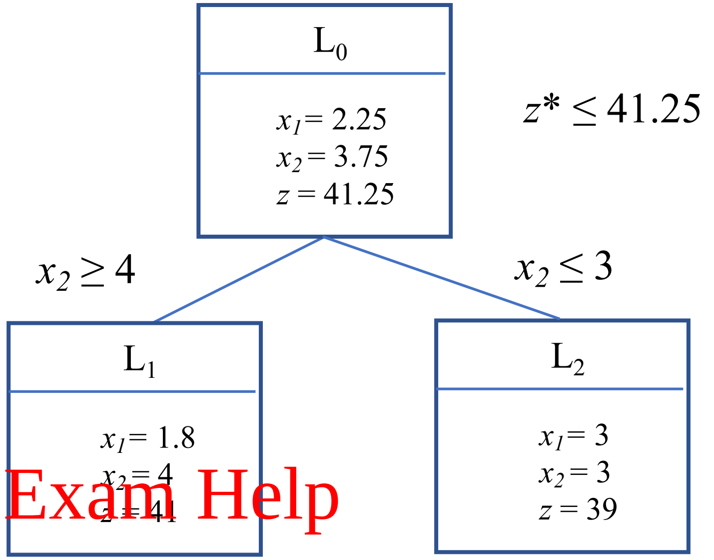


Branch-and-bound summary

- Main idea is to develop bounds $\underline{z} < z^* < \bar{z}$ on z^*
- For maximization problem, lower bound $\underline{z}$ is best integer solution found so far; upper bound $\bar{z}$ is largest LP optimal value at any "hanging" box
- After considering a subregion, we *branch* to split it into two more subregions
- **Three rules for stopping branching on a sub-region:**
  1) LP over subregion L_j is infeasible,
  2) LP solution over subregion L_j gives optimal value *worse* than our best integer solution so far, or
  3) LP solution over subregion L_j is integer-valued.

When we stop branching, this is known as **fathoming** that part of the branch-and-bound tree

Optimality gap



- The **optimality gap** associated with an integer solution is a measure of how far off we are from the true optimal value

- Suppose we stopped the example problem after branching once and solving L_1 and L_2. The upper bound is 41.25, and the lower bound is 39 (because z = 39 corresponds to the best integer solution so far)

- The optimality gap associated with the solution $(x_1,x_2)=(3,3)$ is then

$$\text{Gap} = \frac{\bar{z} - \underline{z}}{\underline{z}} = \frac{41.25 - 39}{39} = 0.057 \ \ (\text{or } 5.7\%)$$

- For large problems where branch-and-bound takes hours to solve, we may prefer to terminate after reaching an optimality gap of 5%, 1%, or 0.1%, etc.

Tractability of integer programming

- Integer programming solvers like Gurobi have improved *dramatically* in recent years, due to advances in optimization theory and algorithms
- Estimated *machine-independent* speedup from 1991 to present (Gurobi 8.0) is a factor of 2 million
- Taking hardware advancements into account, total speedup since 1991 is factor of around 500 *billion*
- Despite computational complexity of integer programming, many models with millions of variables and constraints can now be solved on your laptop
- A caveat: Solution time of IPs is not just a function of # of variables and constraints; some IPs are much harder than others, and may still not be solvable. In general, its hard to predict which IPs will solve fast or slow unless you *try it*
- **Let's see the real estate investment example in Gurobi**

Modeling logical conditions

- A very useful feature of binary variables is the expression of logical conditions in the optimization model
- Consider the real estate development example, where $x_i=1$ if we select project i, and $x_i=0$ if we don't
- Let's look at some examples of logical conditions that frequently arise in integer programming

1. If project 1 is selected, project 5 cannot be selected:

$$x_1 + x_5 \leq 1$$

2. If project 2 is selected, then project 3 must be selected:

$$x_2 \leq x_3$$

Modeling logical conditions

3. You must selected project 1 or 4, or both:

$$x_1 + x_4 \geq 1$$

4. Exactly 3 projects must be selected:

$$\sum_{i=1}^{5} x_i = 3$$

Modeling logical conditions

- Careful: These logical conditions are not always straightforward!

5. Project 3 can be selected only if both projects 1 and 2 are selected

$$1 + x_3 \leq x_1 + x_2$$

Incorrect! Why?

$$2x_3 \leq x_1 + x_2$$

Correct!

- We have so far looked at interactions between binary variables only; we can also use binary variables to model more complex interactions

Non-exclusive OR constraints

- Let x_i, $i = 1,2,3,4$ be continuous decision variables, and assume they are bounded, e.g., $0 \leq x_i \leq 100$ for $i = 1,2,3,4$

- Suppose we want *at least* one of the following two constraints to hold

$$2x_1 + x_2 \geq 5$$

$$2x_3 - x_4 \leq 2$$

- We can do this by introducing a binary variable y_1 and a large constant M:

$$2x_1 + x_2 \geq 5 - My_1$$

$$2x_3 - x_4 \leq 2 + M(1 - y_1)$$

- It is important that we pick M to be **large enough** so we don't accidentally "cut off" other feasible solutions; in this example we can select $M = 200$ because we know $0 \leq x_i \leq 100$

Fixed costs

- In many optimization problems, we would like to model “one-time” fixed costs

- Example: Suppose we are deciding which, if any, of three different products to manufacture and sell:

    - Products 1, 2 and 3 earn profit of \$3, \$4, and \$7 per unit, respectively
    - Products 1, 2 and 3 have market demand of 30, 50 and 25 units, respectively (in thousands)
    - Due to labor constraints, we cannot produce more than 60 thousand units of all products combined
    - Selling each product requires an initial investment for research and development and factory equipment. The required investment to be able to make products 1, 2, and 3 are \$15,000, \$40,000 and \$20,000, respectively.

- Formulate an integer program to determine which products to invest in, and how much of each product to manufacture.

Fixed costs

- Let x_i denote the number of units of product i to manufacture
- Let y_i be an auxiliary variable to denote whether we choose to develop product i
- Let M be a large constant

$$
\begin{aligned}
\max_{x_1,x_2,x_3} \quad & 3x_1 + 4x_2 + 7x_3 - 15,000y_1 - 40,000y_2 - 20,000y_3 \quad \} \text{ Profit function} \\
\text{s.t.} \quad & \left.\begin{array}{l} x_1 \le 30,000 \\ x_2 \le 50,000 \\ x_3 \le 20,000 \end{array}\right\} \text{demand constraints} \\
& \left.x_1 + x_2 + x_3 \le 60,000\right\} \text{labor constraint} \\
& \left.\begin{array}{l} x_1 \le My_1 \\ x_2 \le My_2 \\ x_3 \le My_3 \end{array}\right\} \text{production decisions} \\
& x_1, x_2, x_3 \ge 0 \\
& y_1, y_2, y_3 \in \{0, 1\}.
\end{aligned}
$$

- Because the maximum demand for any product is 50,000, we can use $M = 50{,}000$

In-class assignment 1:

Hospital operating room scheduling