Linear Programming, (Mixed) Integer Linear Programming, and Branch & Bound

COMP8620
Lecture 3-4

Thanks to
Steven Waslander (Stanford)
H. Sarper (Thomson Learning)
who supplied presentation material

What Is a Linear Programming Problem?

Example

Giapetto’s, Inc., manufactures wooden soldiers and trains.

Each soldier built:

- Sell for \$27 and uses \$10 worth of raw materials.
- Increase Giapetto’s variable labor/overhead costs by \$14.
- Requires 2 hours of finishing labor.
- Requires 1 hour of carpentry labor.

Each train built:

- Sell for \$21 and used \$9 worth of raw materials.
- Increases Giapetto’s variable labor/overhead costs by \$10.
- Requires 1 hour of finishing labor.
- Requires 1 hour of carpentry labor.

What Is a Linear Programming Problem?

Each week Giapetto can obtain:

- All needed raw material.
- Only 100 finishing hours.
- Only 80 carpentry hours.

Also:

- Demand for the trains is unlimited.
- At most 40 soldiers are bought each week.

Giapetto wants to maximize weekly profit (revenues – expenses). Formulate a mathematical model of Giapetto's situation that can be used maximize weekly profit.

What Is a Linear Programming Problem?

Decision Variables

x_1 = number of soldiers produced each week

x_2 = number of trains produced each week

Objective Function In any linear programming model, the decision maker wants to maximize (usually revenue or profit) or minimize (usually costs) some function of the decision variables. This function to maximized or minimized is called the objective function.

For the Giapetto problem, fixed costs are do not depend upon the the values of x_1 or x_2.

What Is a Linear Programming Problem?

Giapetto's weekly profit can be expressed in terms of the decision variables *x1* and *x2*:

Weekly profit =

weekly revenue – weekly raw material costs – the weekly variable costs

Weekly revenue = $27x_1 + 21x_2$

Weekly raw material costs = $10x_1 + 9x_2$

Weekly variable costs = $14x_1 + 10x_2$

Weekly profit =

$$(27x_1 + 21x_2) - (10x_1 + 9x_2) - (14x_1 + 10x_2) = 3x_1 + 2x_2$$

What Is a Linear Programming Problem?

Thus, Giapetto’s objective is to choose x_1 and x_2 to maximize $3x_1 + 2x_2$.

Giapetto’s objective function is:

$$\boxed{\text{Maximize } z = 3x_1 + 2x_2}$$

What Is a Linear Programming Problem?

Constraints As x_1 and x_2 increase, Giapetto's objective function grows larger. For Giapetto, the values of x_1 and x_2 are limited by the following three constraints:

Constraint 1 Each week, no more than 100 hours of finishing time may be used.

Constraint 2 Each week, no more than 80 hours of carpentry time may be used.

Constraint 3 Because of limited demand, at most 40 soldiers should be produced.

These three constraints can be expressed as

Constraint 1: $2x_1 + x_2 \leq 100$

Constraint 2: $x_1 + x_2 \leq 80$

Constraint 3: $x_1 \leq 40$

$x_1, x_2 \geq 0$

What Is a Linear Programming Problem?

Maximise $3x_1 + 2x_2$

Subject to

$$2\,x_1 + x_2 \leq 100$$

$$x_1 + x_2 \leq 80$$

$$x_1 \leq 40$$

$$x1,\ x2 \geq 0$$

LP

- Has a linear objective function (to be minimized oir maximized)
- Has constraints that limit the degree to which the objective can be pursued.
- Has a feasible region defining valid solutions (may be empty)
- An optimal solution is a feasible solution that results in the largest possible objective function value when maximizing (or smallest when minimizing).

$$\min \; c'x$$

$$\text{subject to} \quad Ax \le b$$

$$Dx = e$$

$$x \ge 0$$

x	n x 1
c	n x 1
A	m_1 x n
b	m_1 x 1
D	m_2 x n
e	m_2 x n

Standard form of LP

- A linear program is in standard form when
  - The objective is a minimization,
  - all the variables are non-negative , and
  - all other constraints are equalities.
- Multiply maximization objectives by -1 to make a minimization
- Add slack variables to ≤ constraints,
- Subtract surplus variables from ≥ constraints.
- Slack and surplus variables represent the difference between the left and right sides of the original constraints.
- Slack and surplus variables have objective function coefficients equal to 0 (they do not affect the objective function).

Standard form of LP

$$\begin{array}{ll} \text{min} & cx \\ \text{s.t.} & Ax = b \\ & Dx \leq e \\ & Fx \geq g \\ & x \geq 0 \end{array} \quad \Longrightarrow \quad \begin{array}{ll} \text{min} & cx \\ \text{s.t.} & Ax = b \\ & Dx + y = e \\ & Fx - z = g \\ & x, y, z \geq 0 \end{array}$$

y are slack variables, *z* surplus

Search in LP

It can be shown that:

- **The feasible region for any LP will be a convex set.**
- **The feasible region for any LP has only a finite number of extreme points.**
- **Any LP that has an optimal solution has an extreme point that is optimal.**

Search in LP

So, for a small number of variables (like 2), you can solve the problem graphically.

Example 2

$$
\begin{array}{llr}
\text{Max} & 5x_1 + 7x_2 & \\
\text{s.t.} & x_1 \le 6 & (1) \\
 & 2x_1 + 3x_2 \le 19 & (2) \\
 & x_1 + x_2 \le 8 & (3) \\
 & x_1 \ge 0 \text{ and } x_2 \ge 0 &
\end{array}
$$

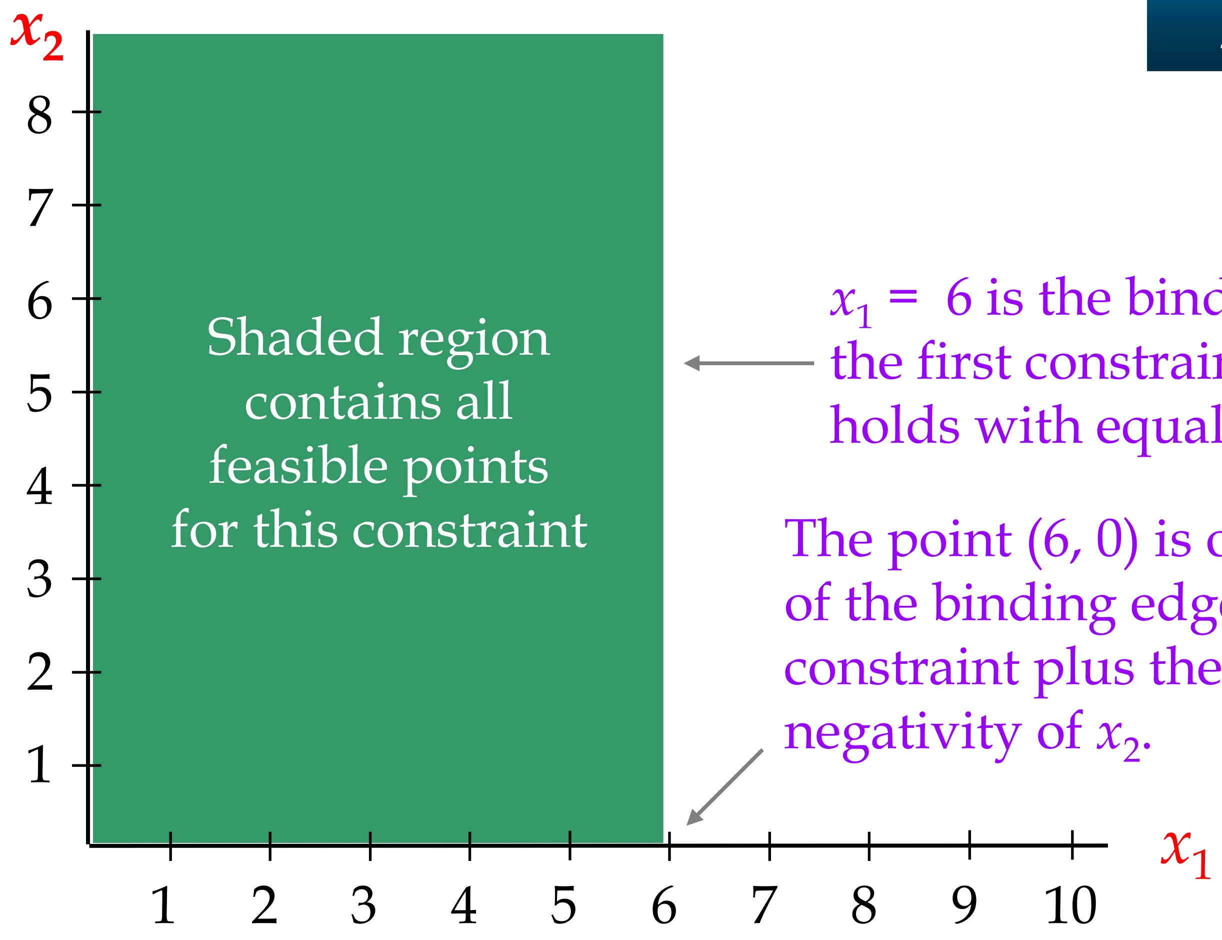
First constraint

Example 2:

Max	$5x_1 + 7x_2$		
s.t.	x_1	$\leq$	6
	$2x_1 + 3x_2$	$\leq$	19
	$x_1 + x_2$	$\leq$	8

$x_1 \geq 0$ and $x_2 \geq 0$

Graph the first constraint of Example 1, plus non-negativity constraints.



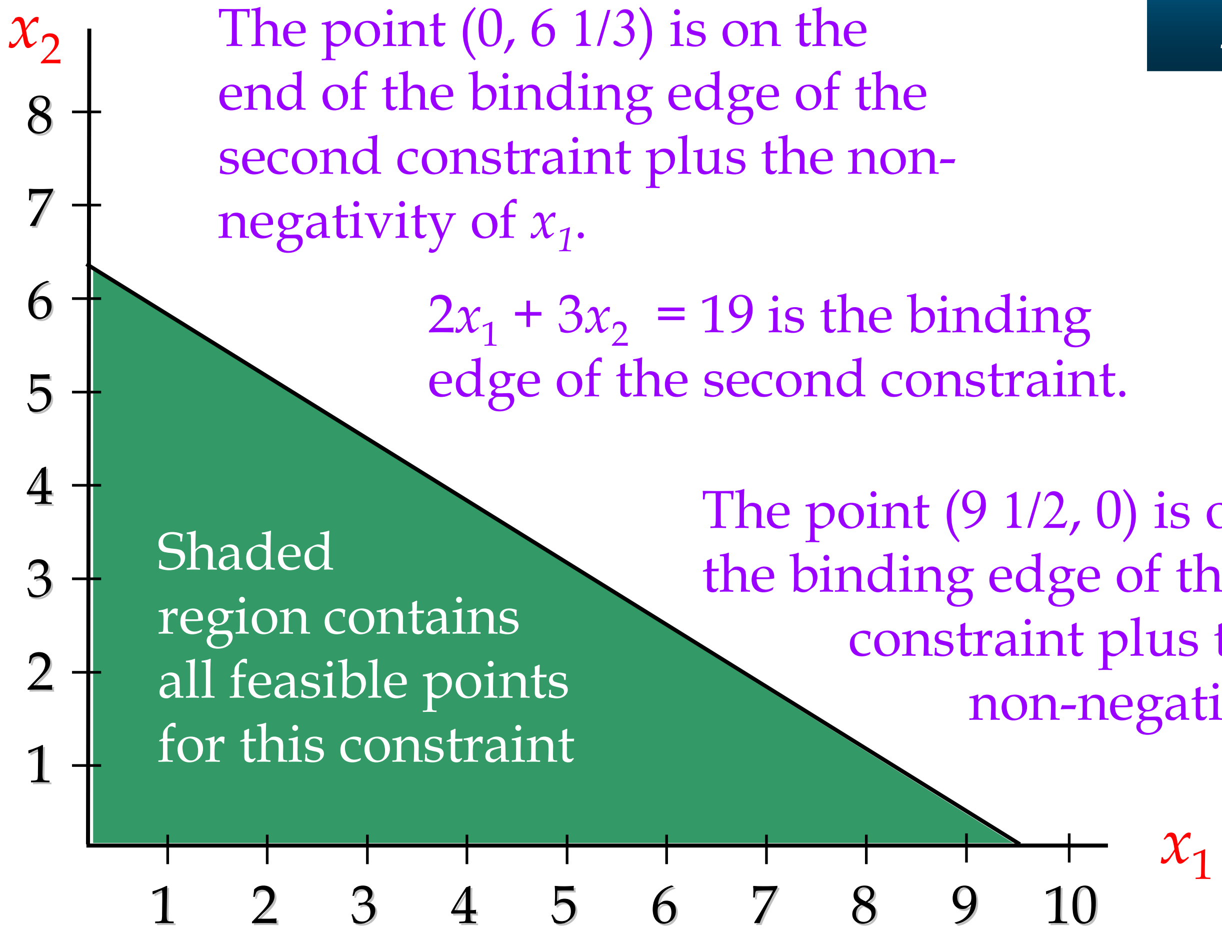
$x_1 = 6$ is the binding edge of the first constraint, where it holds with equality.

The point (6, 0) is on the end of the binding edge of the first constraint plus the non-negativity of x_2.

Graph the second constraint of Example 1, plus non-negativity constraints.

Example 2:

$$\begin{array}{lrcl} \text{Max} & 5x_1 + 7x_2 & & \\ \text{s.t.} & x_1 & \le & 6 \\ & 2x_1 + 3x_2 & \le & 19 \\ & x_1 + x_2 & \le & 8 \\ & x_1 \ge 0 \text{ and } x_2 & \ge & 0 \end{array}$$



The point (0, 6 1/3) is on the end of the binding edge of the second constraint plus the non-negativity of x_1.

$2x_1 + 3x_2 = 19$ is the binding edge of the second constraint.

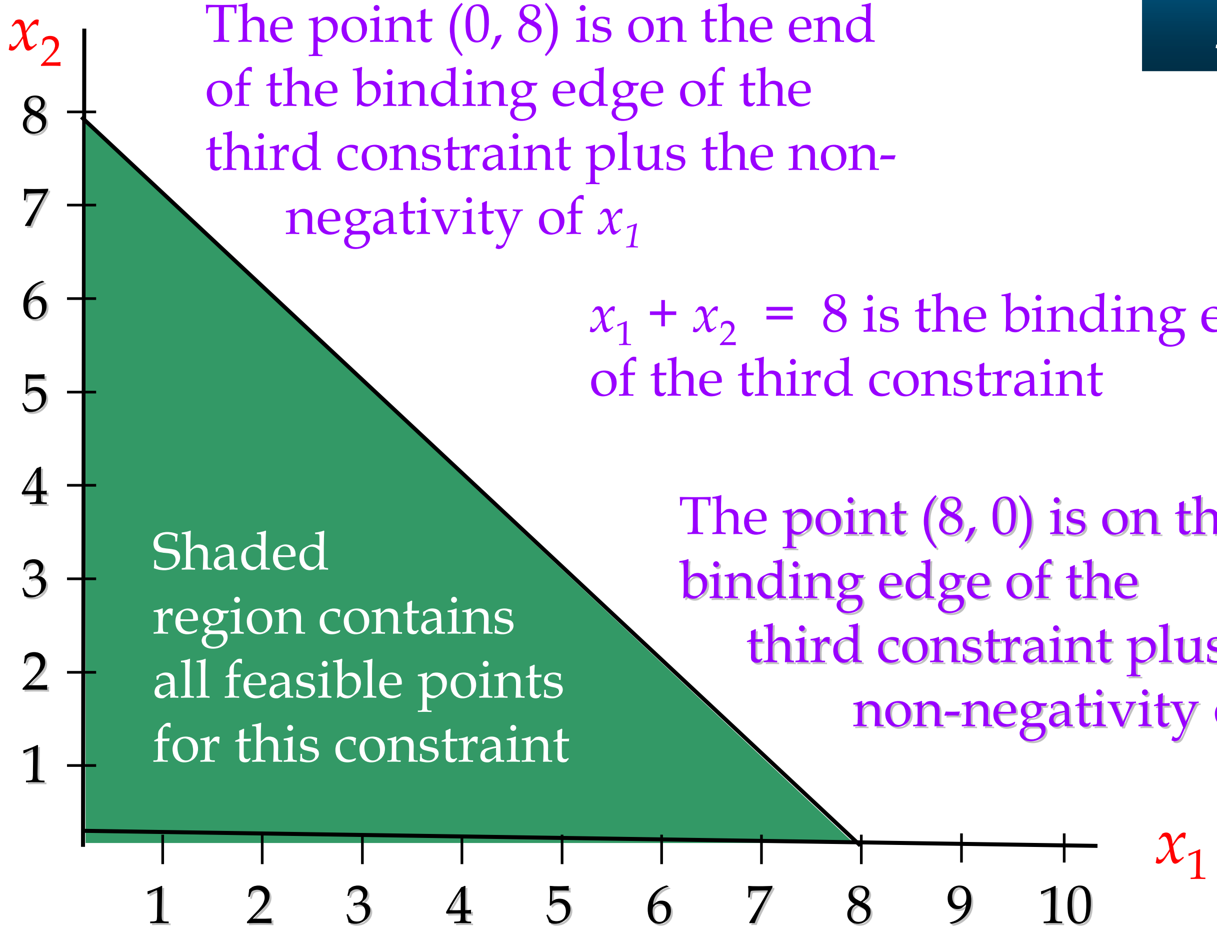
The point (9 1/2, 0) is on the end of the binding edge of the second constraint plus the non-negativity of x_2.

Third constraint

Example 2:

$$\begin{array}{lrcl} \text{Max} & 5x_1 + 7x_2 & & \\ \text{s.t.} & x_1 & \le & 6 \\ & 2x_1 + 3x_2 & \le & 19 \\ & x_1 + x_2 & \le & 8 \\ & x_1 \ge 0 \text{ and } x_2 \ge 0 & & \end{array}$$

Graph the third constraint of Example 1, plus non-negativity constraints.



The point (0, 8) is on the end of the binding edge of the third constraint plus the non-negativity of x_1

$x_1 + x_2 = 8$ is the binding edge of the third constraint

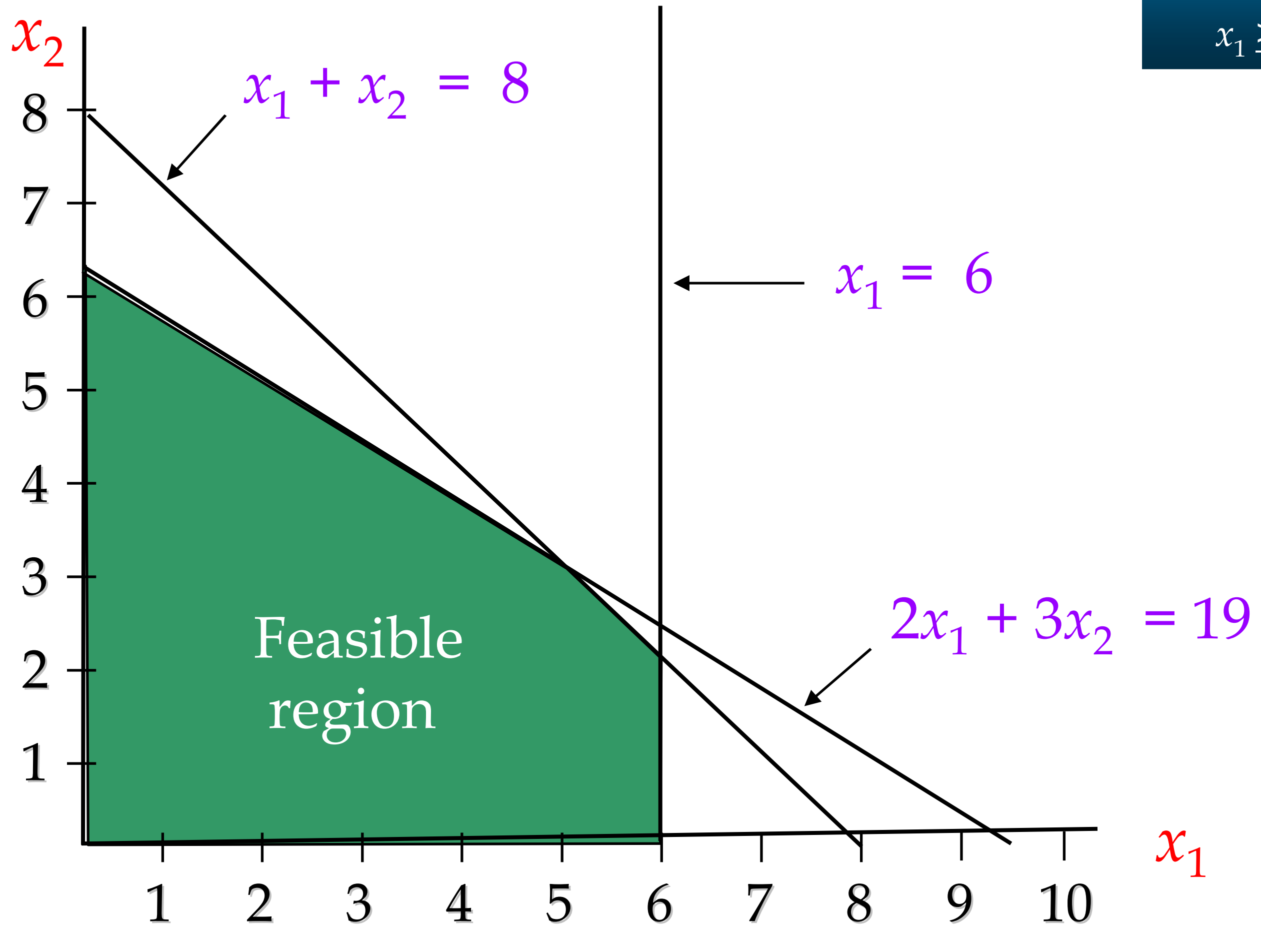
The point (8, 0) is on the end of the binding edge of the third constraint plus the non-negativity of x_2

Example 2:

Max $5x_1 + 7x_2$

s.t.
$$\begin{aligned} x_1 &\le 6 \\ 2x_1 + 3x_2 &\le 19 \\ x_1 + x_2 &\le 8 \\ x_1 \ge 0 \text{ and } x_2 &\ge 0 \end{aligned}$$

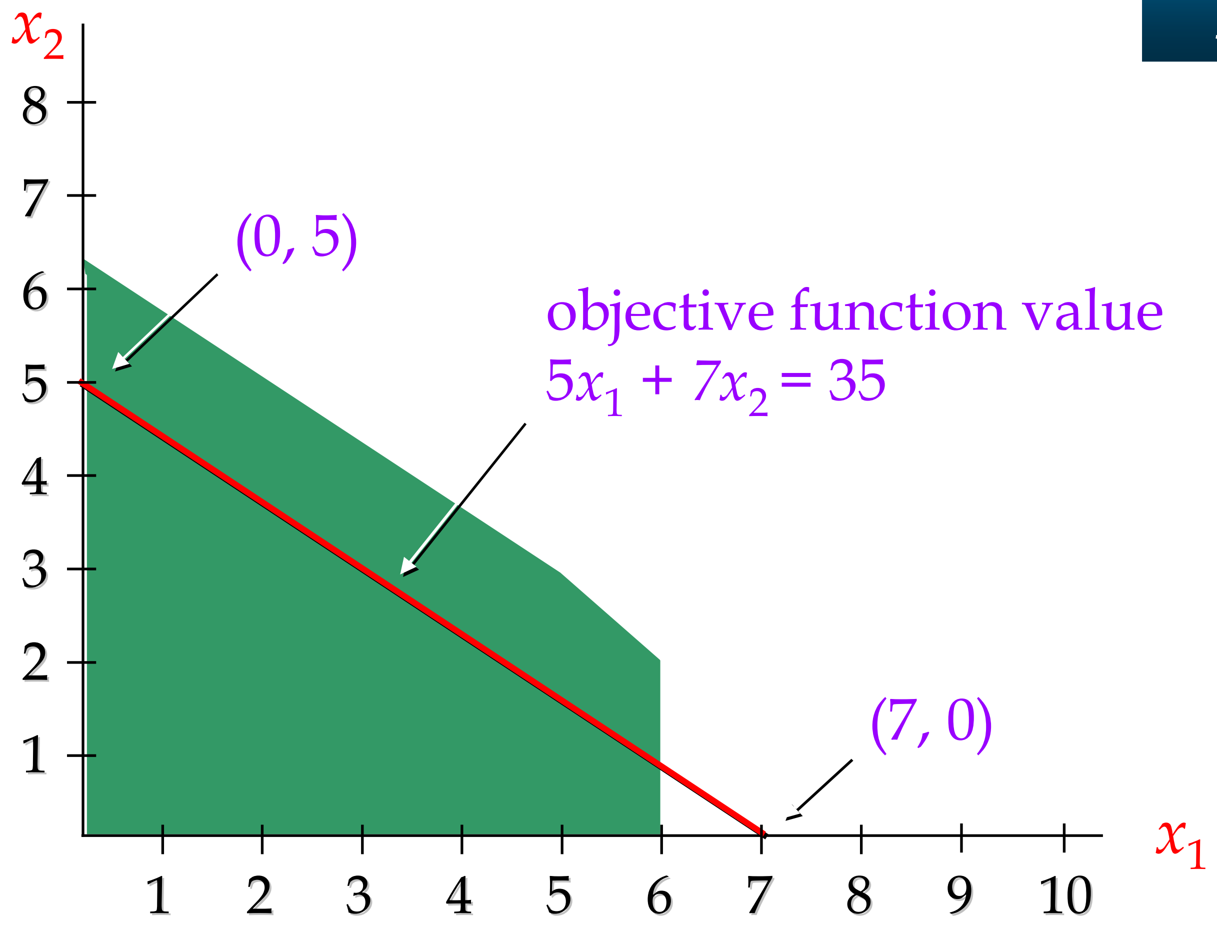
Intersect all constraint graphs to define the feasible region.



Graph a line with a constant objective function value. For example, 35 dollars of profit.

Example 2:

Max	$5x_1 + 7x_2$		
s.t.	x_1	$\leq$	6
	$2x_1 + 3x_2$	$\leq$	19
	$x_1 + x_2$	$\leq$	8
	$x_1 \geq 0$ and $x_2 \geq 0$		



Alternative constant-value lines

Graph alternative constant-value lines.
For example, 35 dollars, 39 dollars, or 42 dollars of profit.

Example 2:

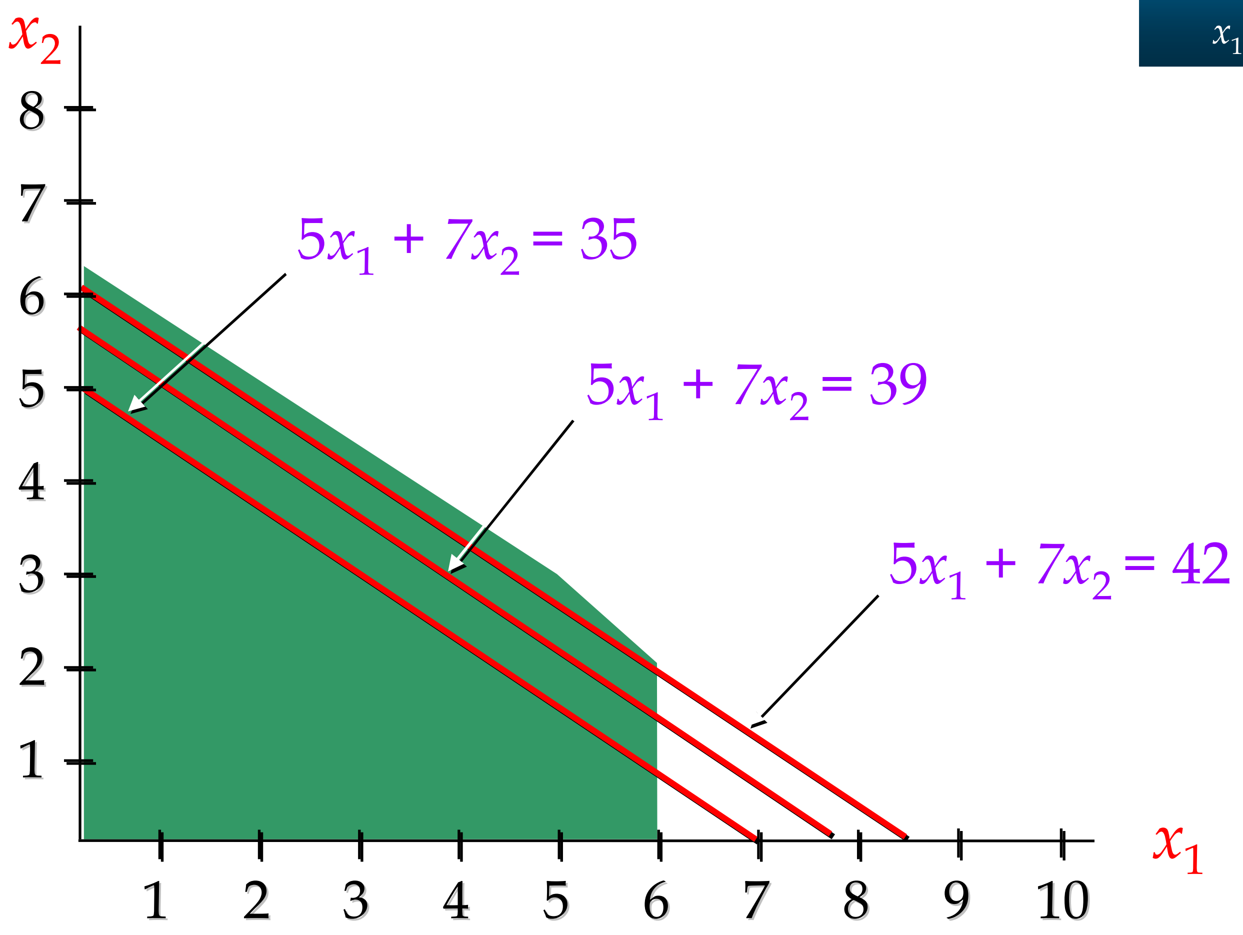
Max $5x_1 + 7x_2$

s.t. $x_1 \leq 6$

$2x_1 + 3x_2 \leq 19$

$x_1 + x_2 \leq 8$

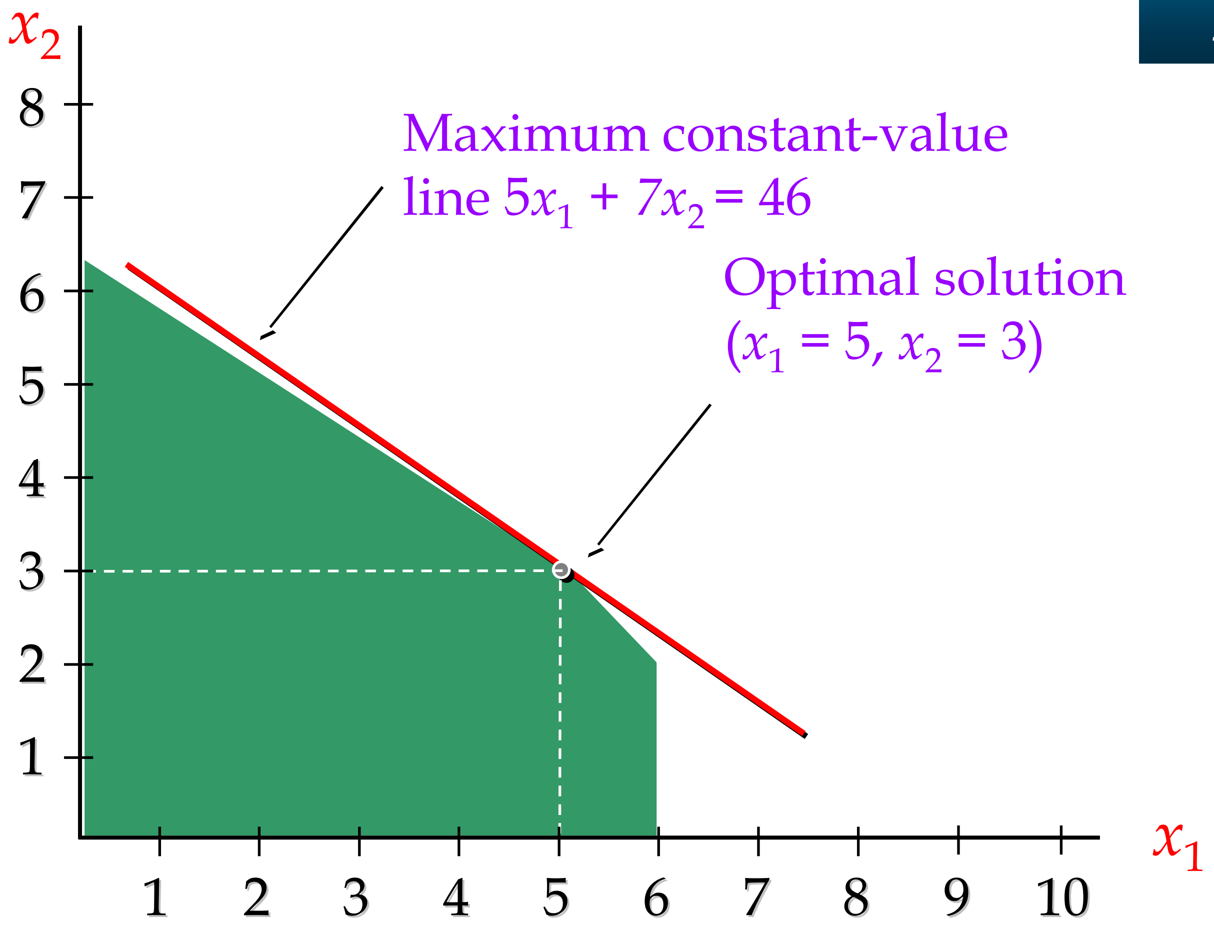
$x_1 \geq 0$ and $x_2 \geq 0$



Graph the maximum constant-value line, graph the optimal solution, then estimate coordinates.

Example 2:

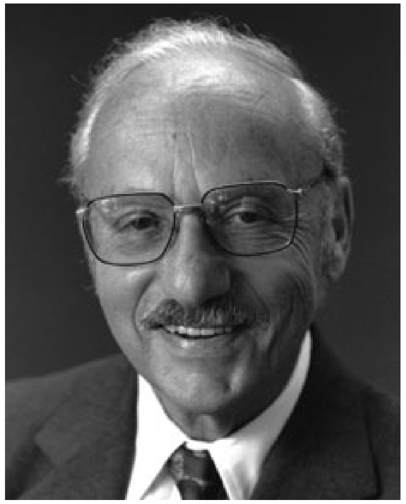
$$\begin{array}{lrcl} \text{Max} & 5x_1 + 7x_2 & & \\ \text{s.t.} & x_1 & \leq & 6 \\ & 2x_1 + 3x_2 & \leq & 19 \\ & x_1 + x_2 & \leq & 8 \\ & x_1 \geq 0 \text{ and } x_2 \geq 0 & & \end{array}$$



Solution spaces

- Example 2 had a unique optimum
- If objective is parallel to a constraint, there are infinite solutions
- If feasible region is empty, there is no solution
- If feasible region is infinite, the solution may be unbounded.

Solving LP



(Dantzig 1951) **Simplex method**

- Very efficient in practice
- Exponential time in worst case

(Khachiyan 1979) **Ellipsoid method**

- *Not* efficient in practice
- Polynomial time in worst case

Solving LP

- Simplex method operates by visiting the extreme points of the solution set
- If, in standard form, the problem has m equations in n unknowns $(m < n)$, setting $(n - m)$ variables to 0 give a basis of m variables, and defines an extreme point.
- At each point, it moves to a neighbouring extreme point by moving one variable into the basis (makes value > 0) and moving one out of the basis (make value 0)
- It moves to the neighbour that *apparently* increases the objective the most (heuristic)
- If no neighbour increases the objective, we are done

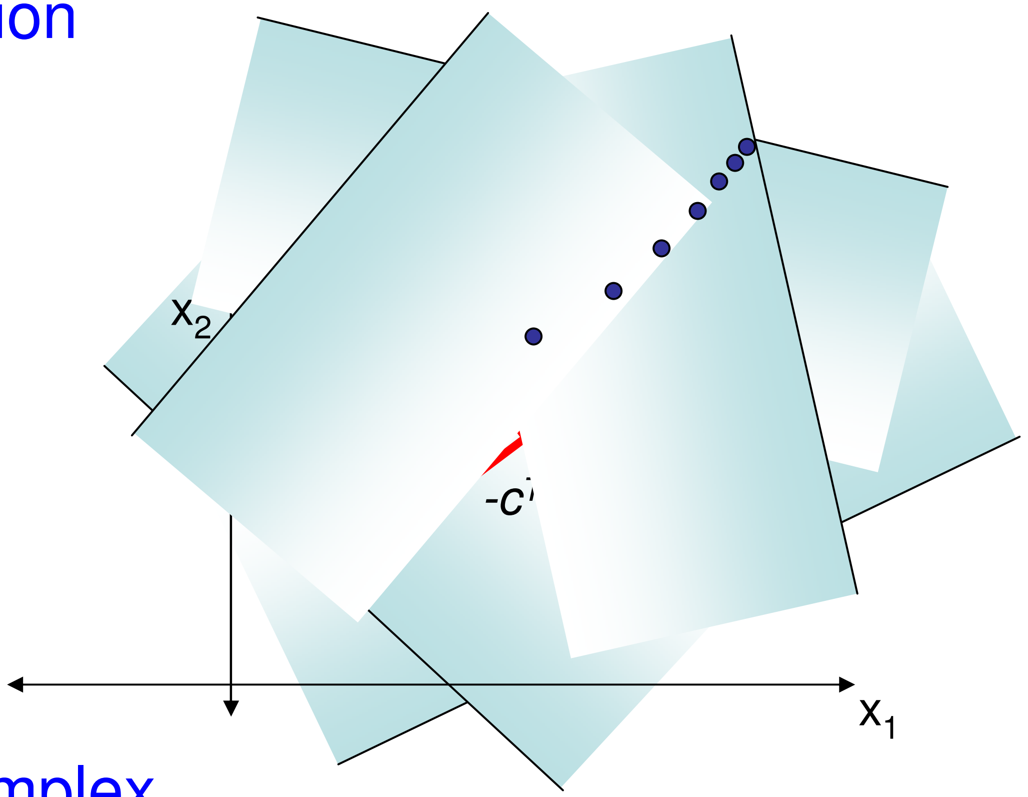
Solving LP

Problems in solving using Simplex:

- Basic variables with value 0
- Rounding (numerical instability)
- Degeneracy (many constraints intersecting in a small region, so each step moves only a small distance)

Solving LP

- Interior Point Methods
  - Apply Barrier Function to each constraint and sum
  - Primal-Dual Formulation
  - Newton Step



- Benefits
  - Scales Better than Simplex
  - Certificate of Optimality

Solving LP

- Variants of the Simplex method exist
  - e.g. Network Simplex for solving flow problems

- Problems with thousands of variables and thousands of constraints are routinely solved (even a few million variables if you only have low-thousands of constraints)

- Or vice-versa

Solving the LP

- Simplex method is a “Primal” method: it stays feasible, and moves toward optimality
- Other methods are “Dual” methods: they maintain an optimal solution toa relaxed problem, and move toward feasibility.

Solving LP

- Everyone uses commercial software to solve LPs
- Basic method for a few variables available in Excel
- ILOG CPLEX is world leader
- Xpress-MP from Dash Optimization is also very good
- Several others in the marketplace
- lp_solve open source project is very useful

Using LP

- LP requires
  - Proportionality: The contribution of the objective function from each decision variable is proportional to the value of the decision variable.
  - Additivity: The contribution to the objective function for any variable is independent of the other decision variables
  - Divisibility**:** each decision variable be permitted to assume fractional values
  - Certainty: each parameter (objective function coefficients, right-hand side, and constraint coefficients) are known with certainty

Using LP

The Certainty Assumption & Sensitivity analysis

- For each decision variable, the shadow cost (aka reduced cost) tells what the benefit from changes in the value around the optimal value
- Tells us which constraints are binding at the optimum, and the value of relaxing the constraint

Beyond LP

Linear Programming sits within a hierarchy of mathematical programming problems

General Optimization Program

- Standard form:

$$\boxed{\begin{array}{lll} \min\limits_{x \in X} & f(x) & \\ \text{s.t.} & g(x) & \le 0 \\ & h(x) & = 0 \end{array}}$$

where

$$X \text{ can be anything}$$

$$f, g, h : X \to \mathbb{R}$$

- Too general to solve, must specify properties of *X*, *f*,*g* and *h* more precisely.

Diversion… Complexity Analysis

- **(P)** – Deterministic **P**olynomial time algorithm
- **(NP)** – **N**on-deterministic **P**olynomial time algorithm,
  - Feasibility can be determined in polynomial time
- **(NP-complete)** – **NP** and at least as hard as any known **NP** problem
- **(NP-hard)** – not provably **NP** and at least as hard as any **NP** problem,
  - Optimization over an **NP-complete** feasibility problem

Optimization Problem Types – Real Variables

- Linear Program (LP)
  - (P) Easy, fast to solve, convex

$$\begin{array}{ll} \min\limits_{x\in\mathbb{R}^n} & c^T x \\ \text{s.t.} & Ax \leq b \end{array}$$

- Non-Linear Program (NLP)
  - (P) Convex problems easy to solve
  - Non-convex problems harder, not guaranteed to find global optimum

$$\begin{array}{ll} \min\limits_{x\in\mathbb{R}^n} & f(x) \\ \text{s.t.} & g(x) \leq 0 \\ & h(x) = 0 \\ \text{where,} & f, g, h : \mathbb{R}^n \to \mathbb{R} \end{array}$$

Optimization Problem Types – Integer/Mixed Variables

- Integer Programs (IP) : $X \subseteq \mathbb{Z}^n$
  - (NP-hard) computational complexity
- Mixed Integer Linear Program (MILP)
  - Generally (NP-hard)

$$\boxed{\begin{array}{rc} \min\limits_{x\in X} & c^T x \\ \text{s.t.} & Ax \leq b \\ \text{where,} & X \subseteq \mathbb{Z}^{n_i} \times \mathbb{R}^{n_r} \end{array}}$$

  - However, many problems can be solved surprisingly quickly!

(Mixed) Integer Programming

- Integer Programming: all variables must have Integer values
- Mixed Integer Programming : some variables have integer values

Exponential solution times

Integer Programming

Example IP formulation:

The Knapsack problem:

$$\text{Let } x_i = \begin{cases} 1 & \text{if I select item } i \\ 0 & \text{otherwise} \end{cases}$$

I wish to select items to put in my backpack.

- There are *m* items available.
- Item *i* weights w_i kg,
- Item *i* has value v_i.
- I can carry Q kg.

$$\max \quad \sum_{\text{i}} x_i v_i$$

$$\text{s.t.} \quad \sum_{\text{i}} x_i w_i \le Q$$

$$x_i \in \{0,1\}$$

Integer Programming

- IP allows formulation "tricks"
  e.g. If *x* then not *y*:

$$(1 - x)\,M \geq y$$

  (*M* is "big M" – a large value – larger than any feasible value for *y*)

Solving ILP

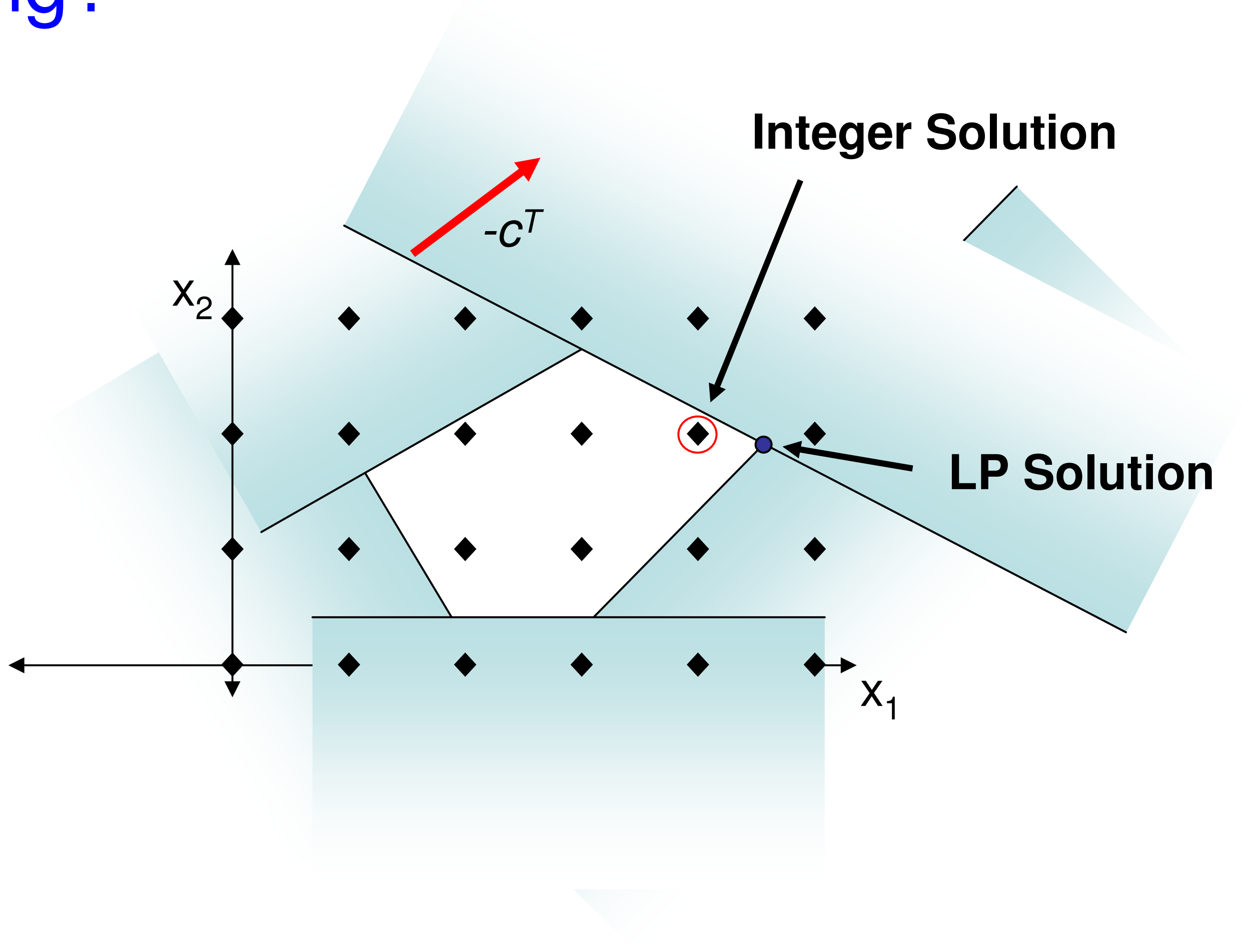
How can we solve ILP problems?

Solving ILP

- Some problem classes have the “Integrality Property”: All solution naturally fall on integer points
- e.g.
  - Maximum Flow problems
  - Assignment problems
- If the constraint matrix has a special form, it will have the Integrality Property:
  - Totally Unimodular
  - Balanced
  - Perfect

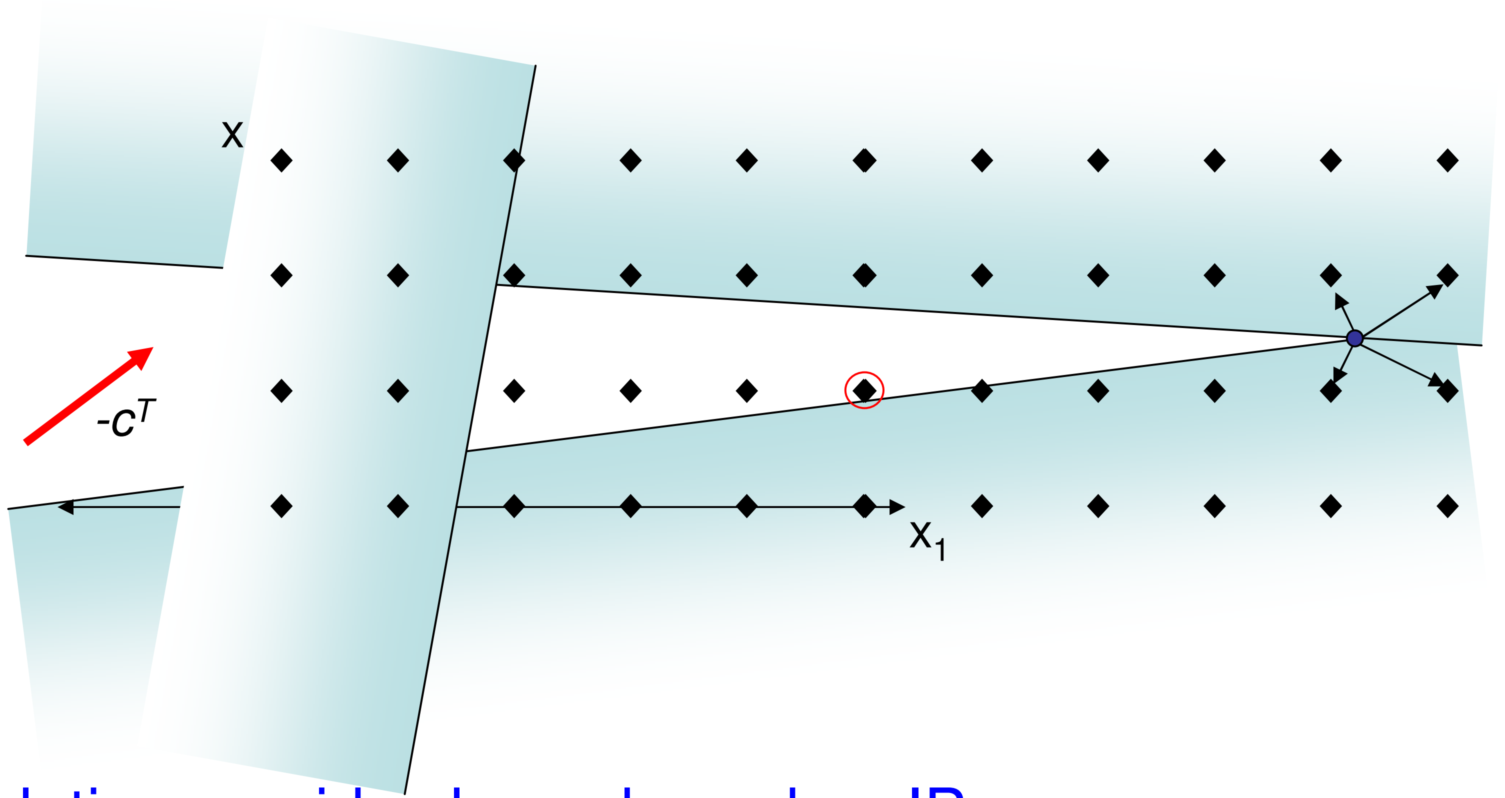
Solving ILP

- How about solving LP Relaxation followed by rounding?



Solving ILP

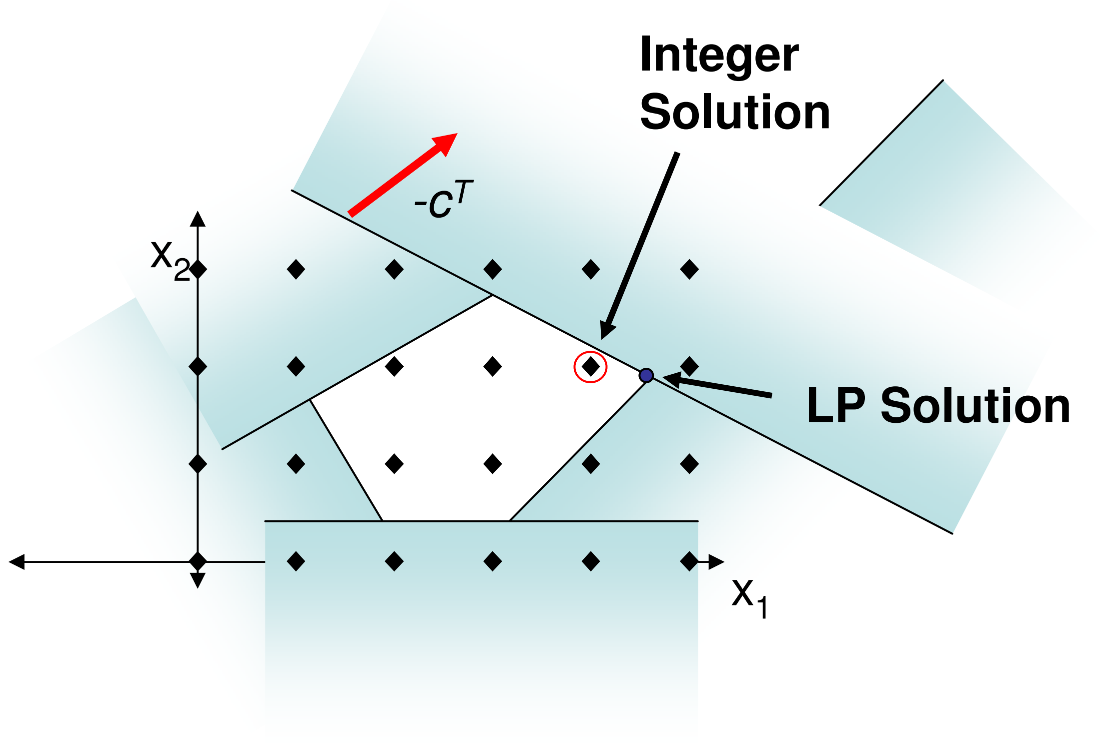
- In general, though, it don’t work



- LP solution provides lower bound on IP
- But, rounding can be arbitrarily far away from integer solution

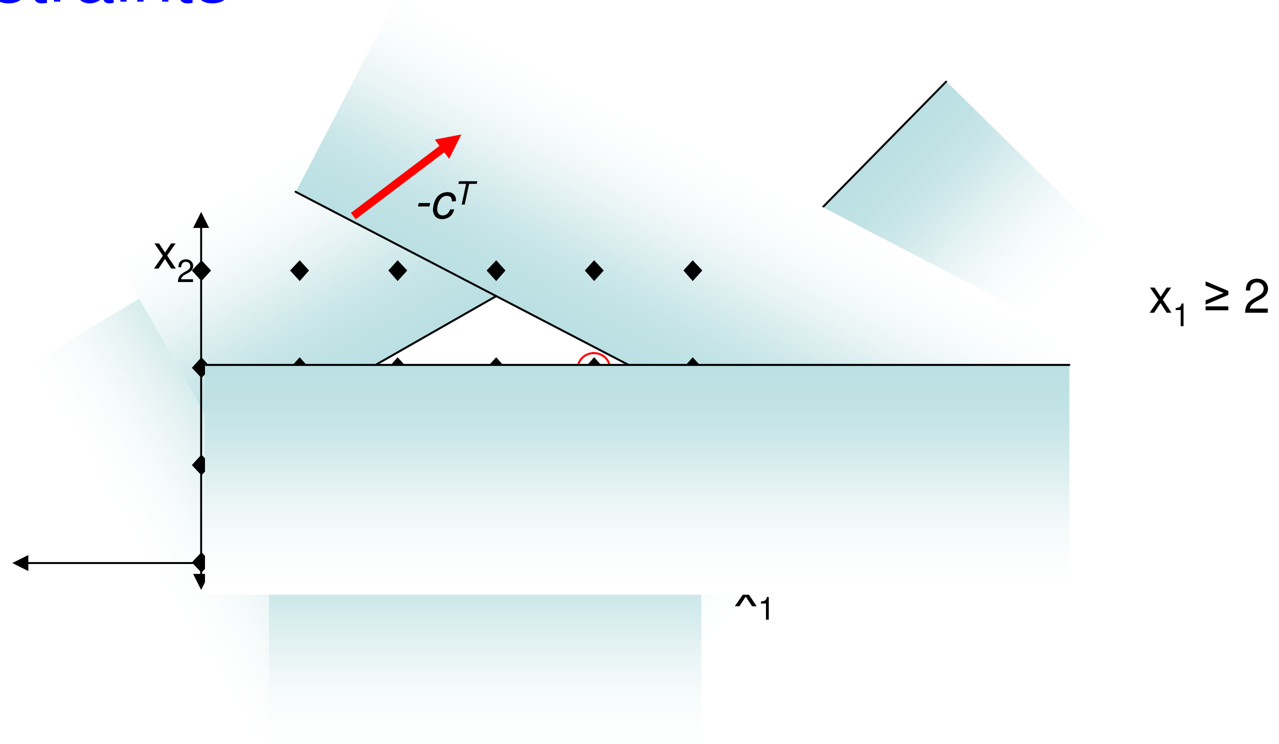
Solving ILP

- Combine both approaches
  - Solve LP Relaxation to get fractional solutions
  - Create two sub-branches by adding constraints



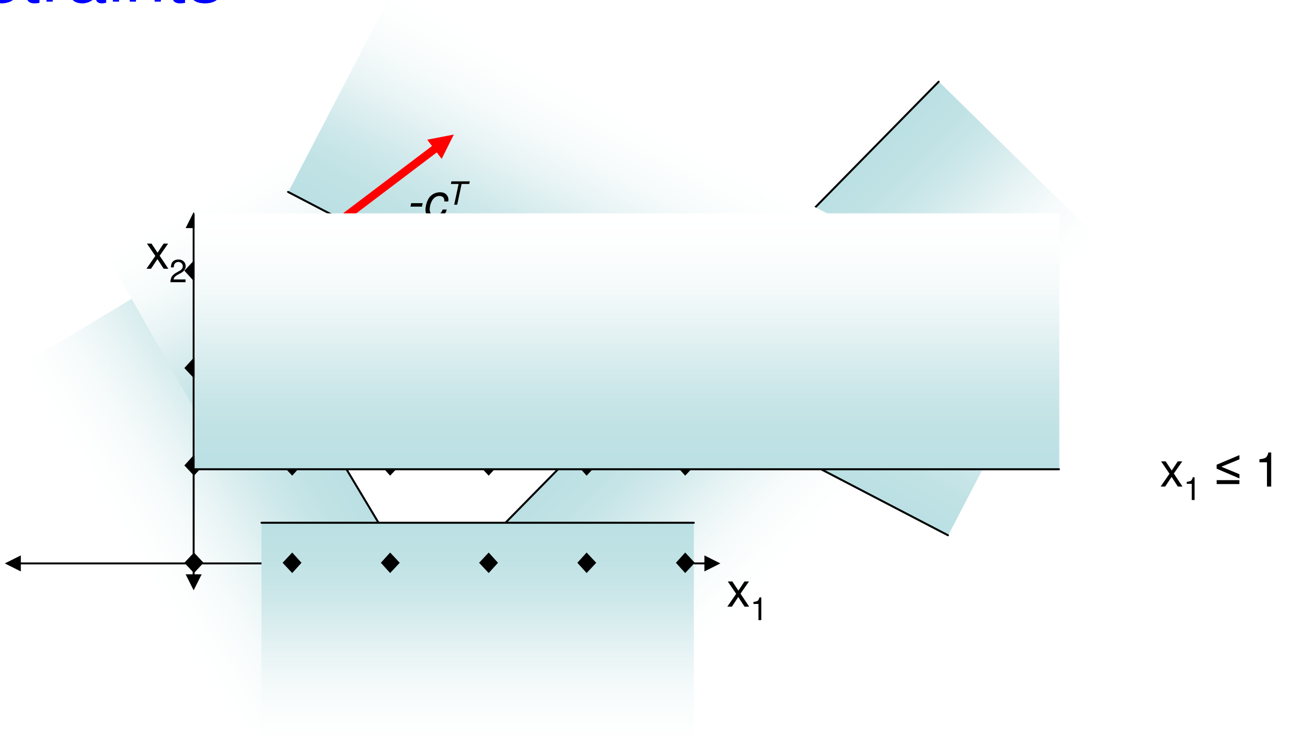
Solving ILP

- Combine both approaches
  - Solve LP Relaxation to get fractional solutions
  - Create two sub-branches by adding constraints



Solving ILP

- Combine both approaches
  - Solve LP Relaxation to get fractional solutions
  - Create two sub-branches by adding constraints



Branch & Bound

- Branch and Bound Algorithm
  1. Solve LP relaxation for lower bound on cost for current branch
     - If solution exceeds upper bound, branch is terminated
     - If solution is integer, replace upper bound on cost
  2. Create two branched problems by adding constraints to original problem
     - Select integer variable with fractional LP solution
     - Add integer constraints to the original LP
  3. Repeat until no branches remain, return optimal solution.

Branch & Bound

- Example: Problem with 4 variables, all required to be integer

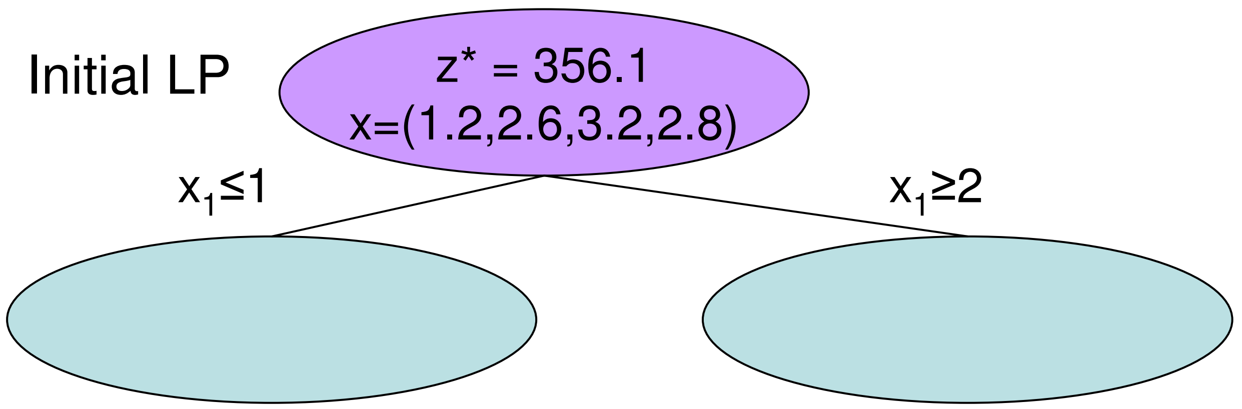
Branch & Bound

Initial LP

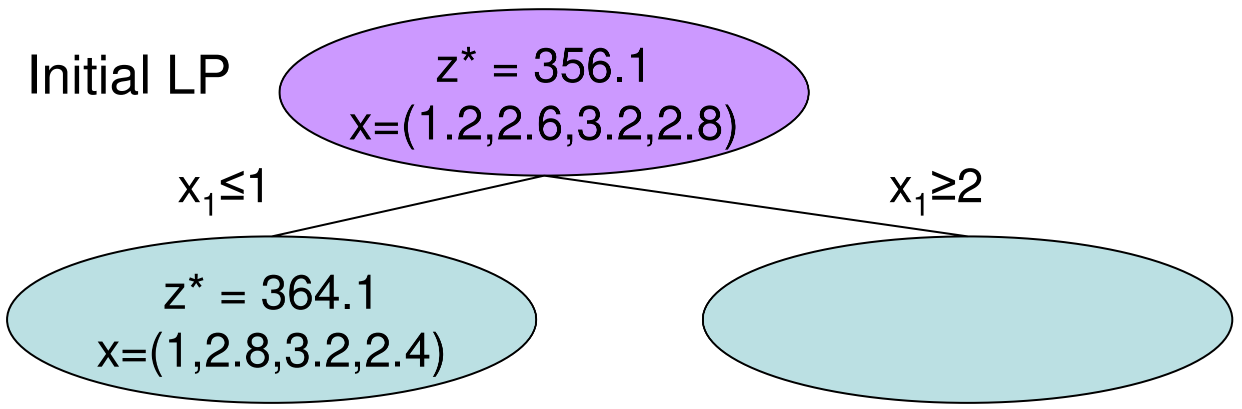
$z^* = 356.1$

$x=(1.2,2.6,3.2,2.8)$

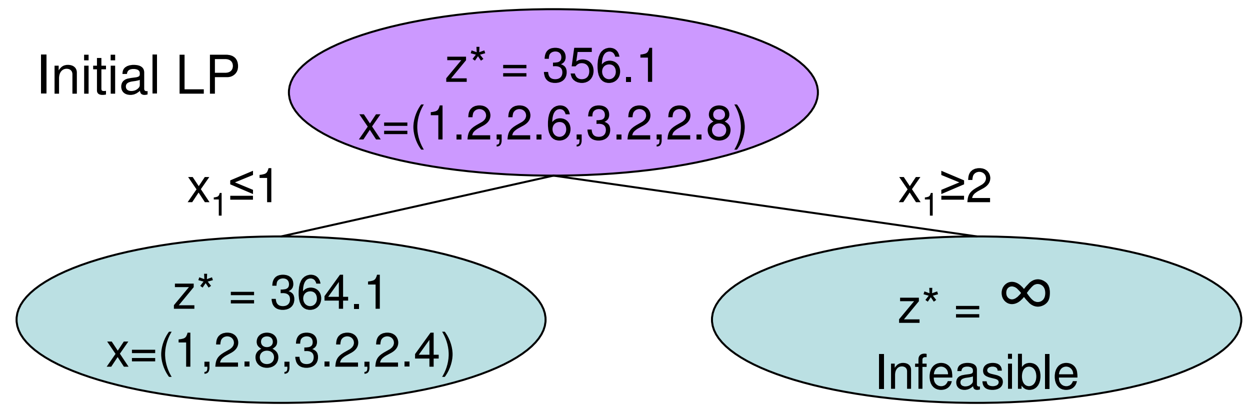
Branch & Bound



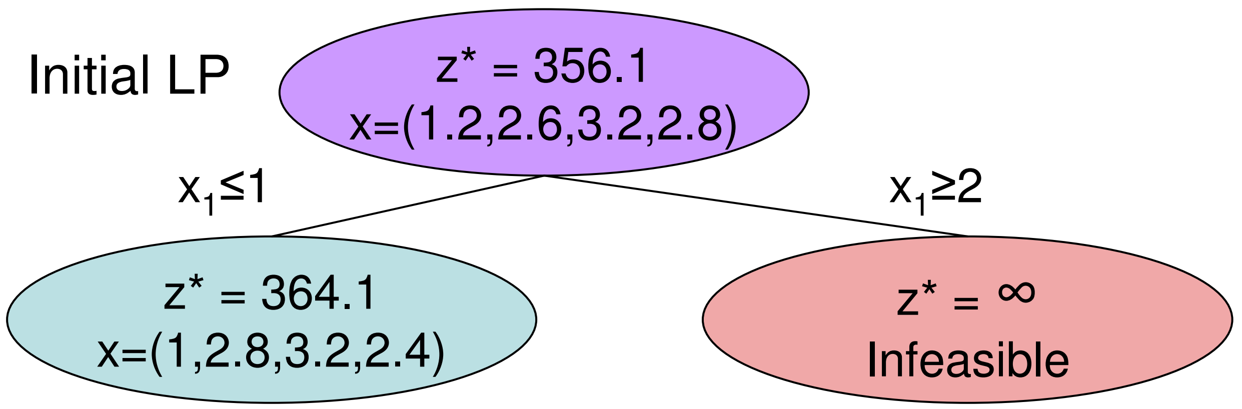
Branch & Bound



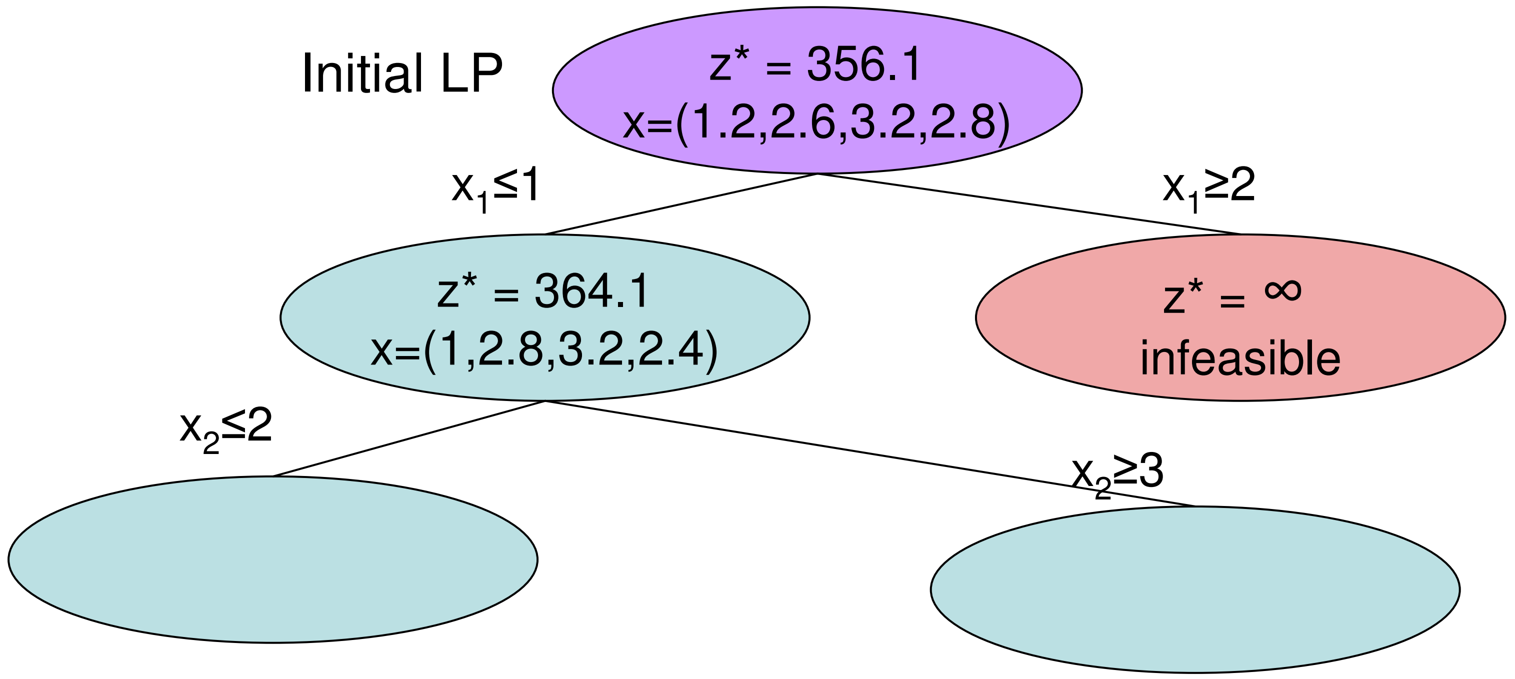
Branch & Bound



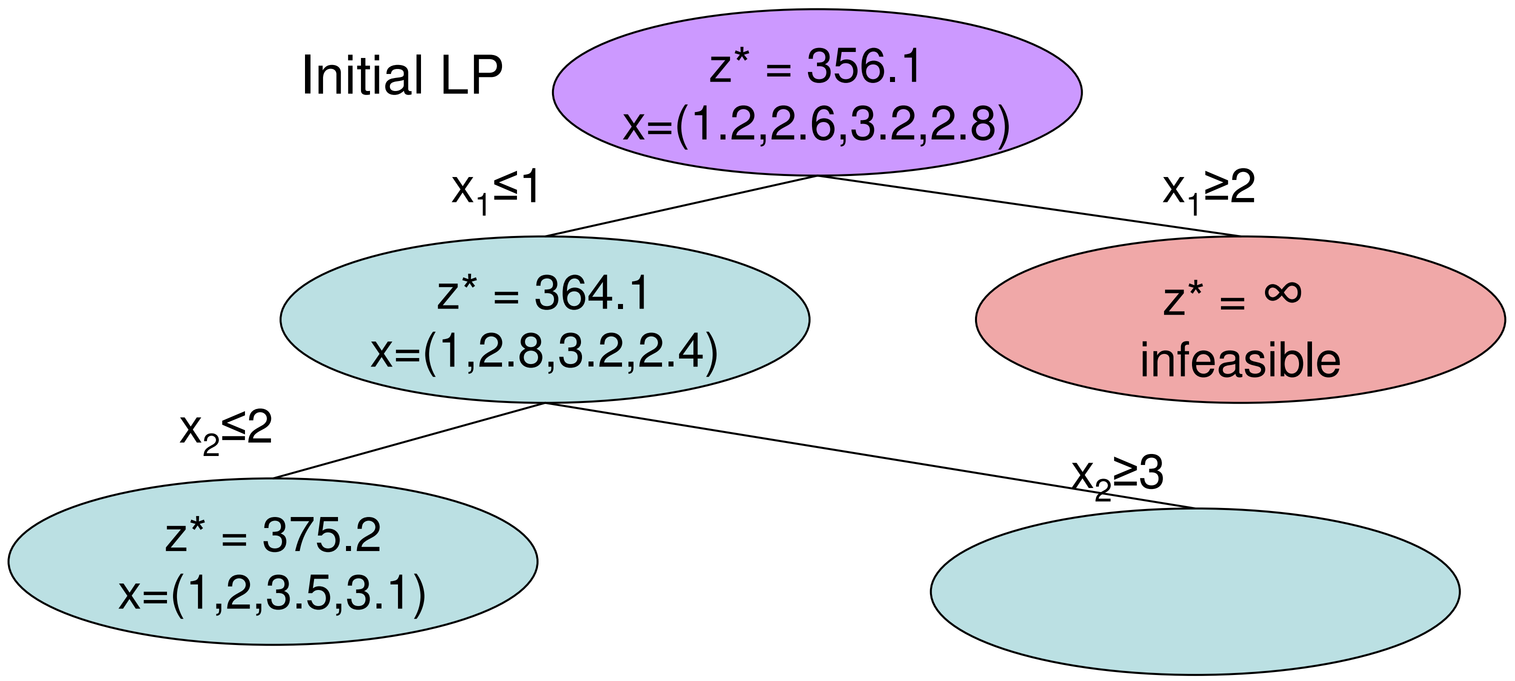
Branch & Bound



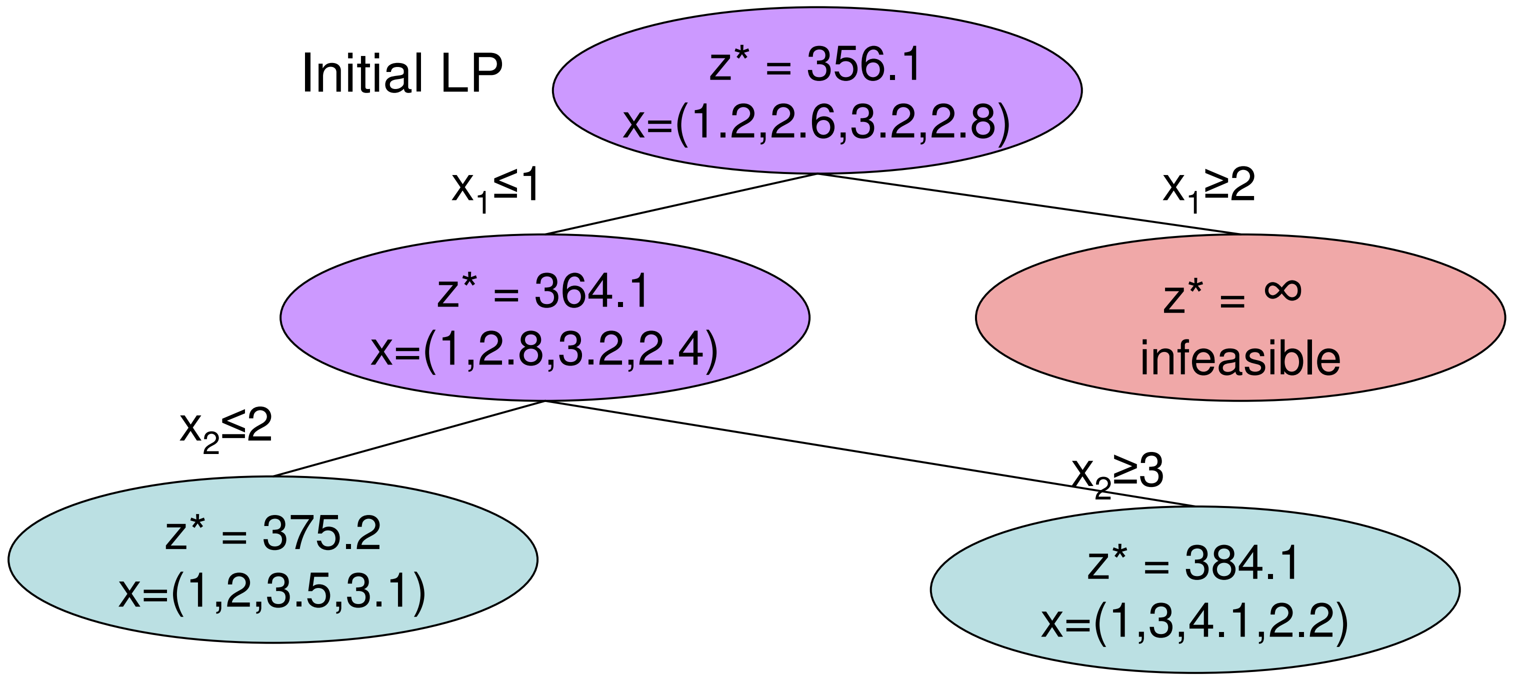
Branch & Bound



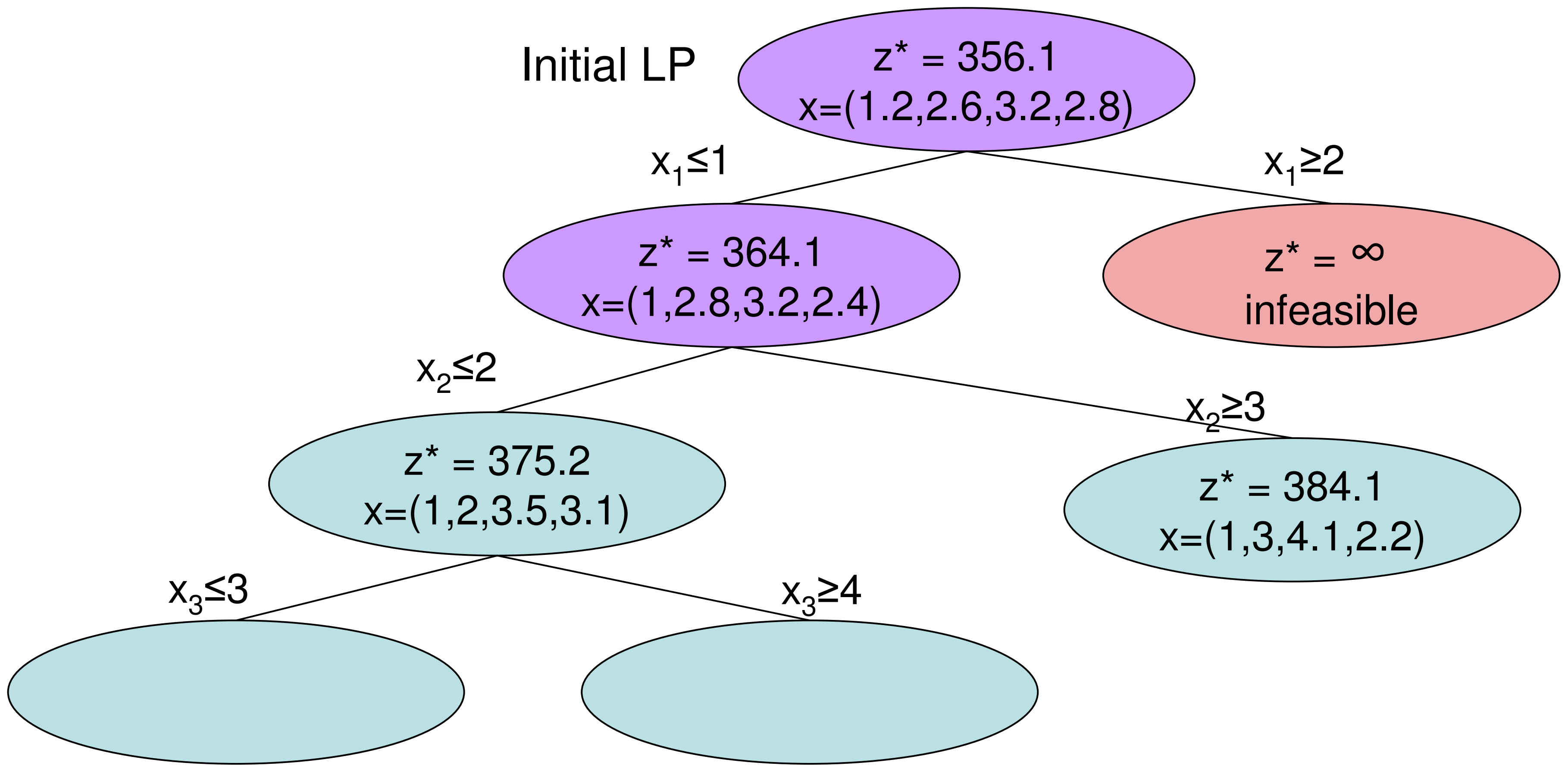
Branch & Bound



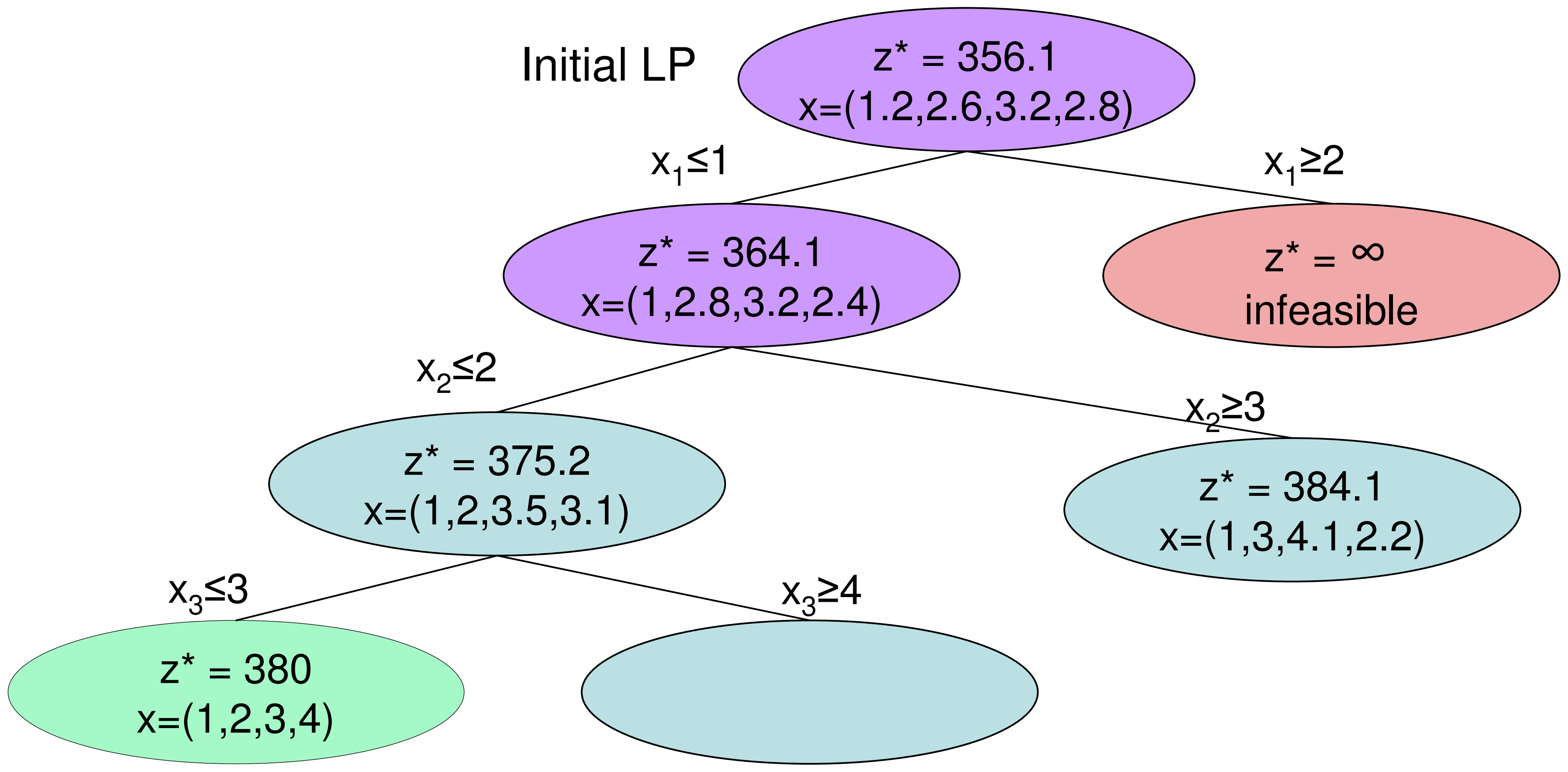
Branch & Bound



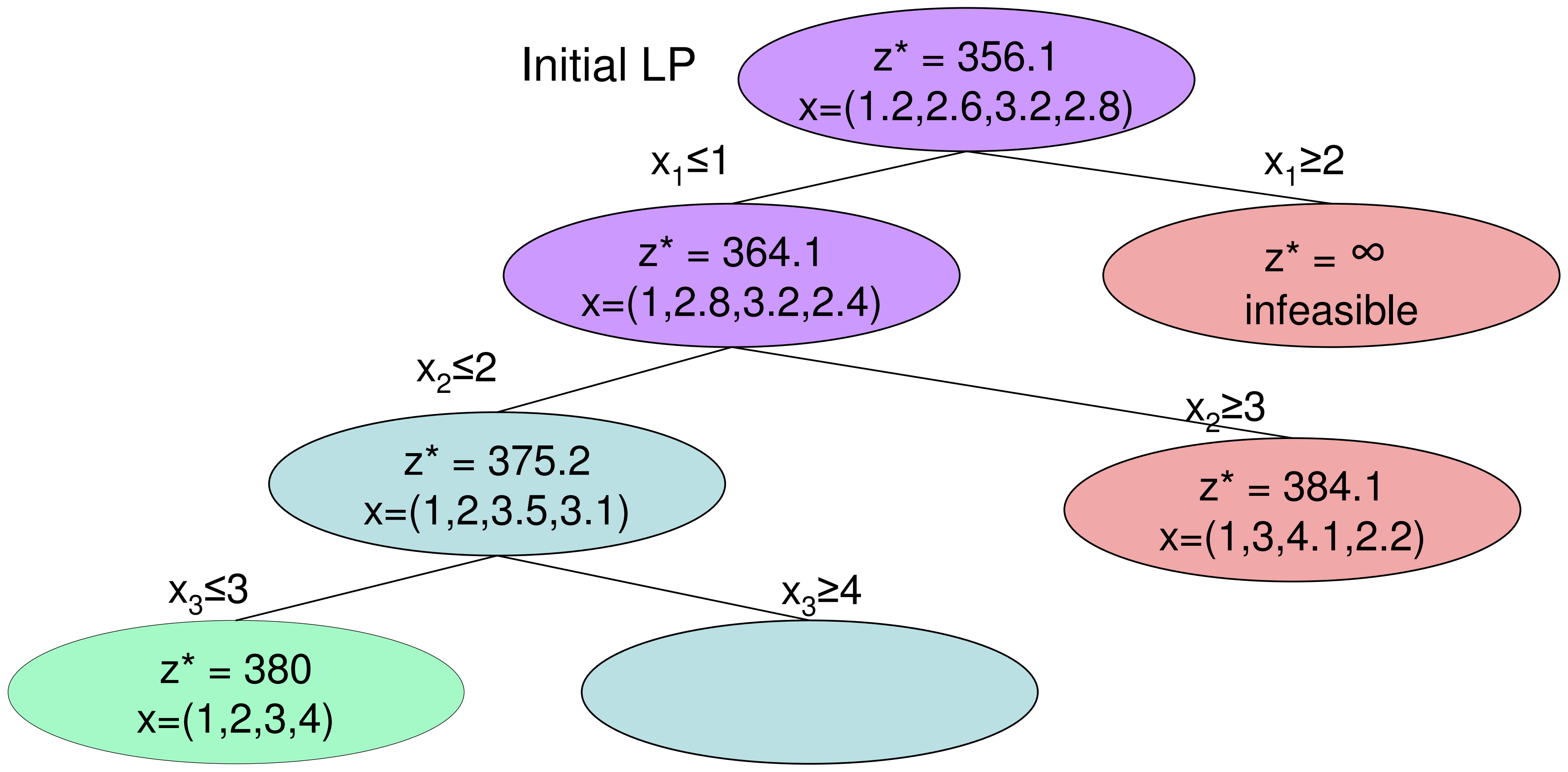
Branch & Bound



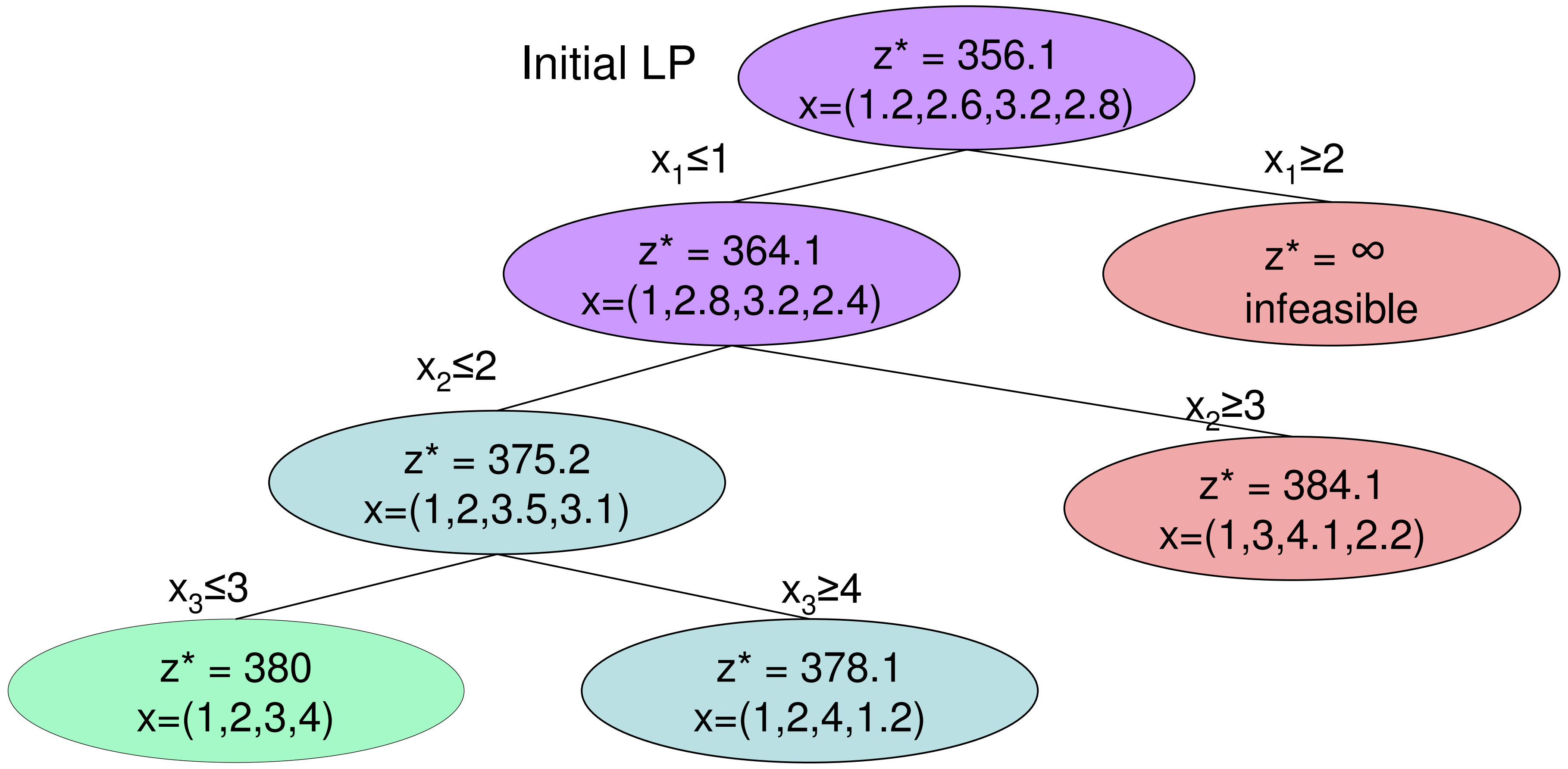
Branch & Bound



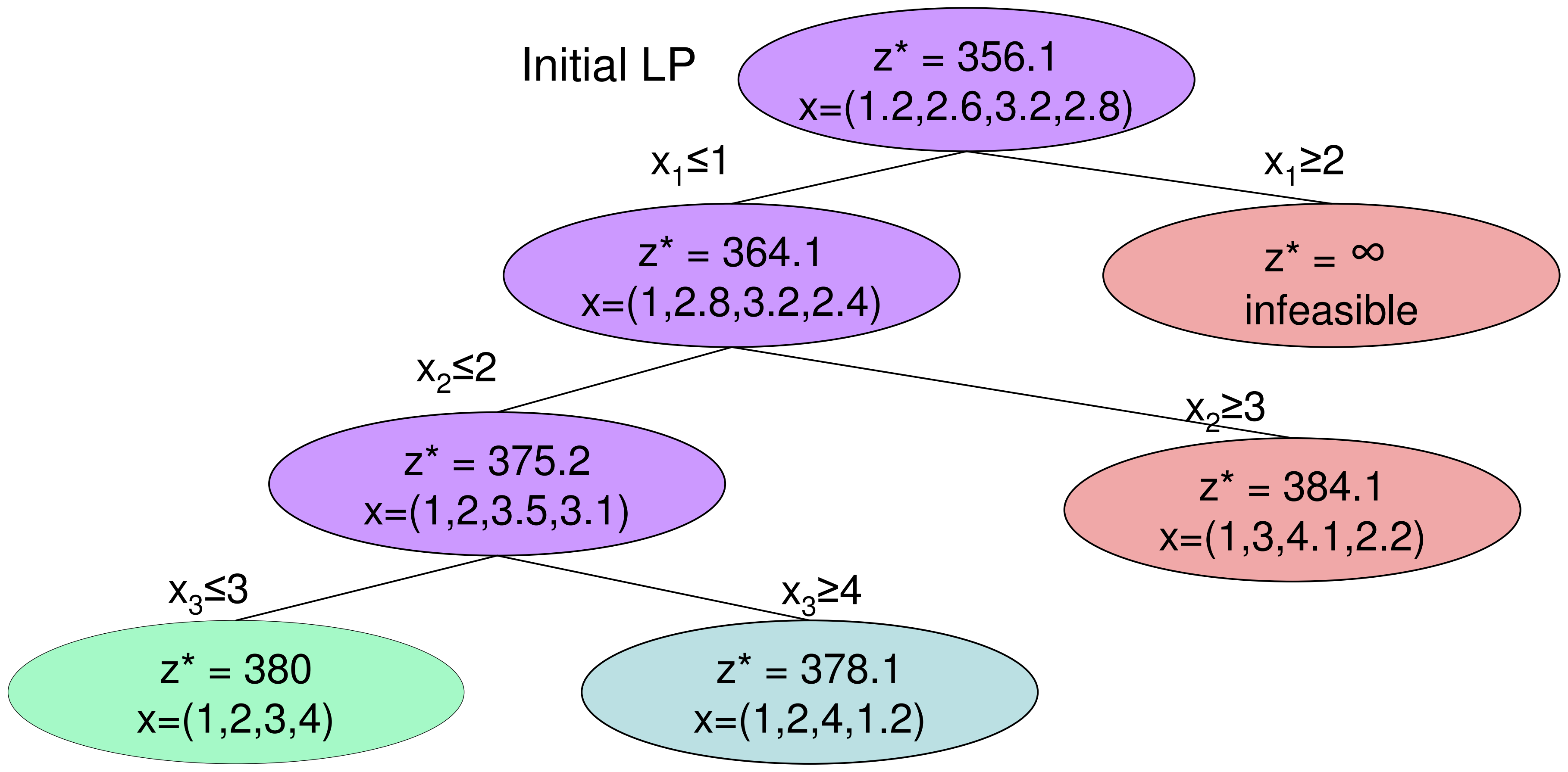
Branch & Bound



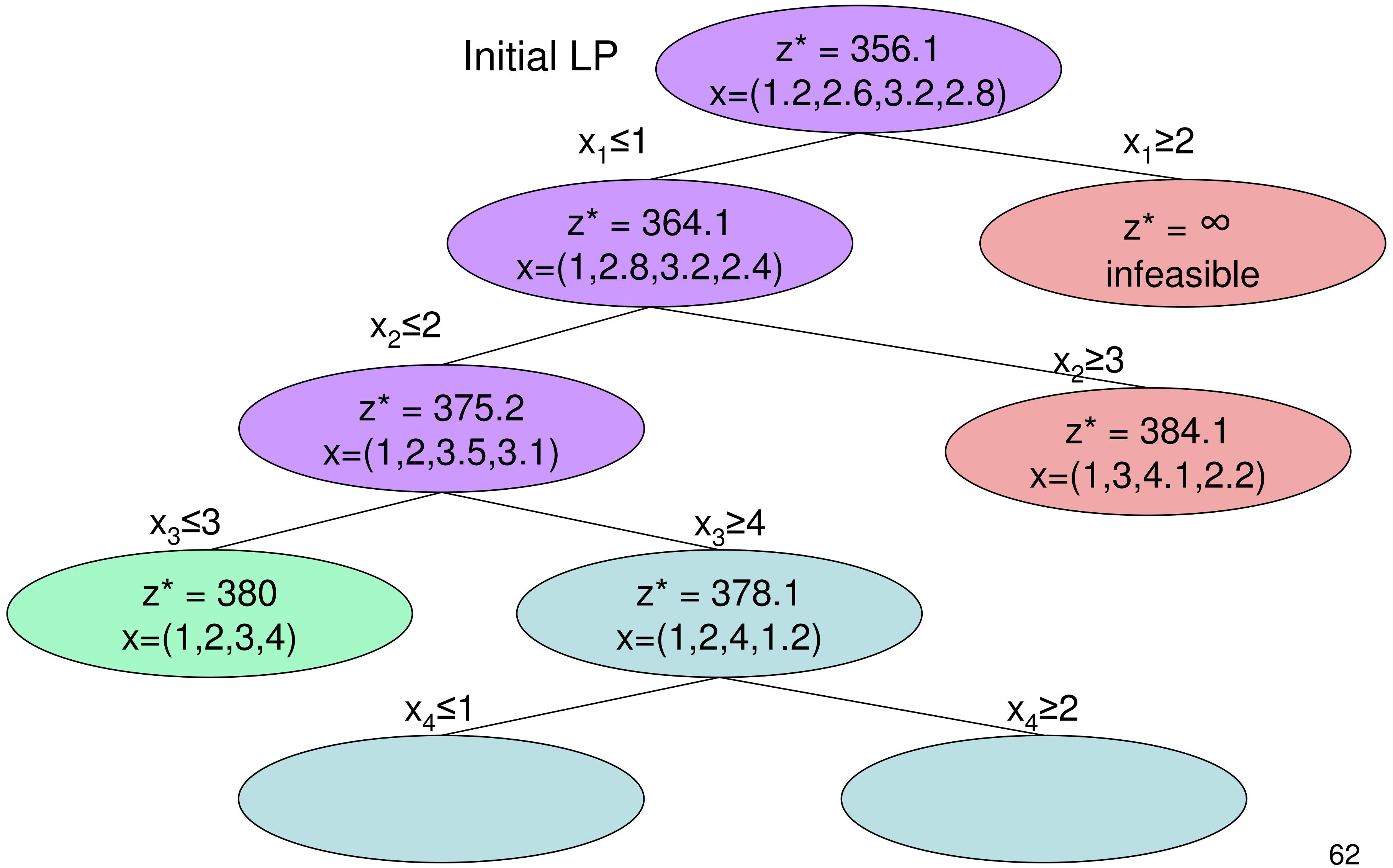
Branch & Bound



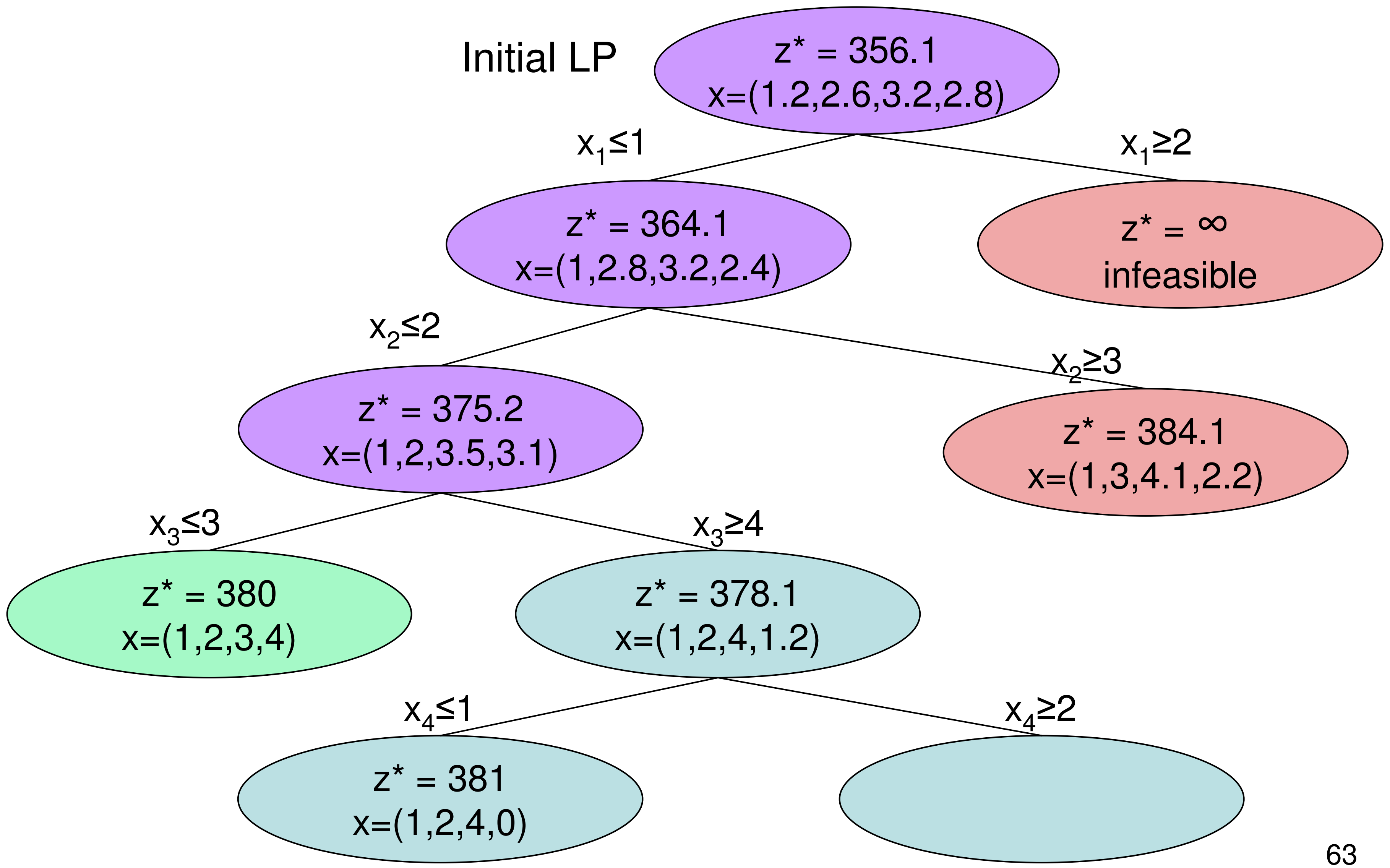
Branch & Bound



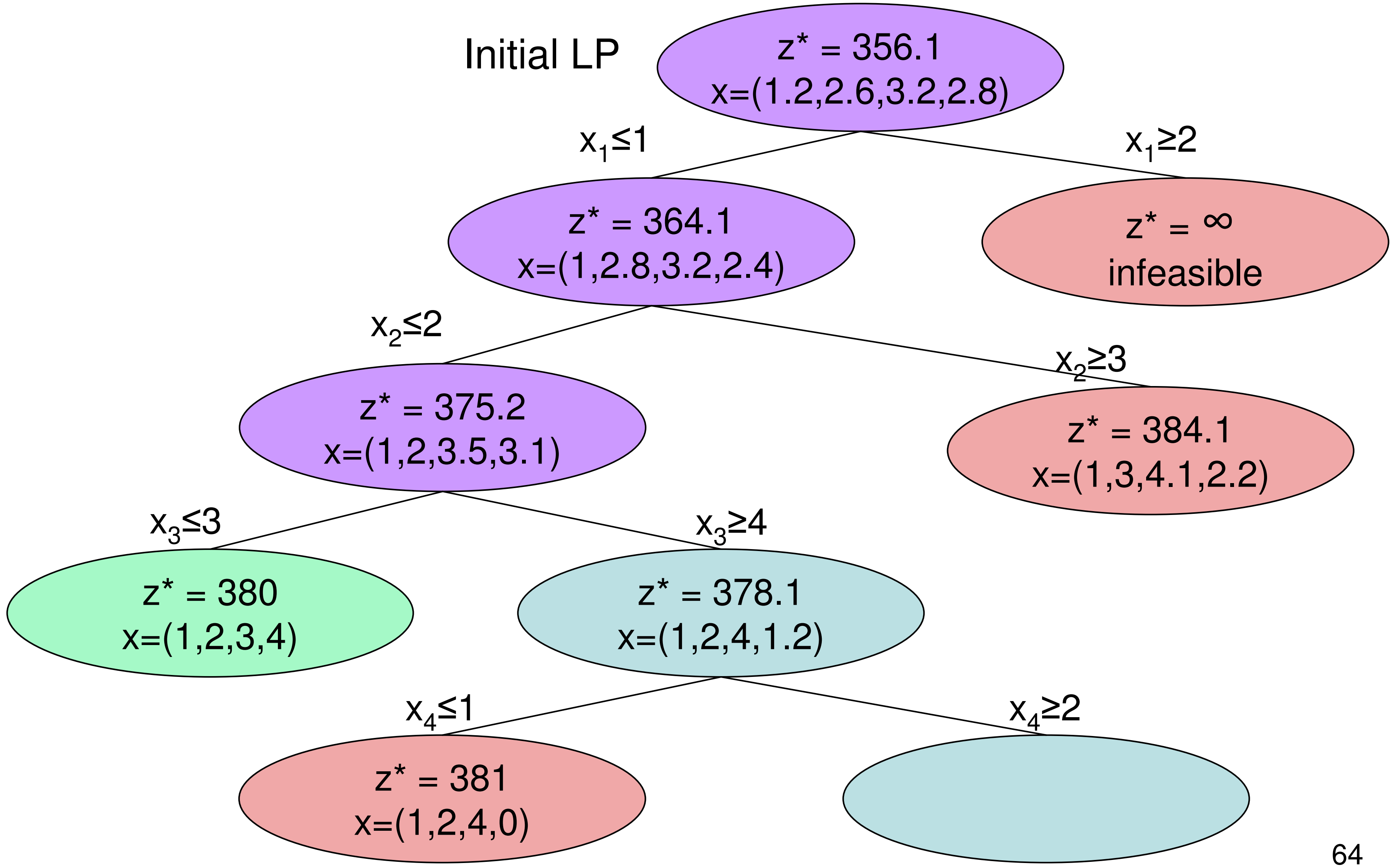
Branch & Bound



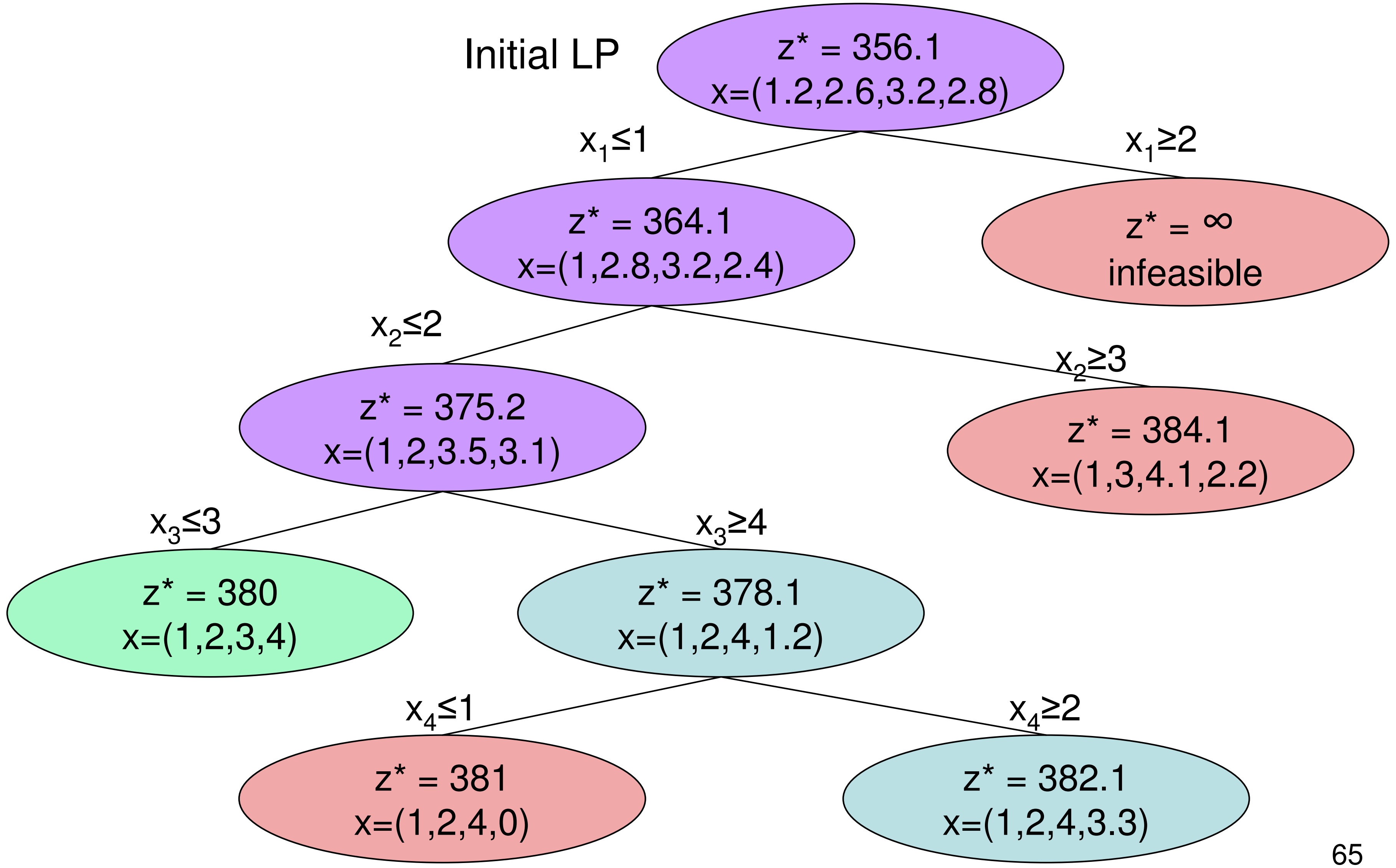
Branch & Bound



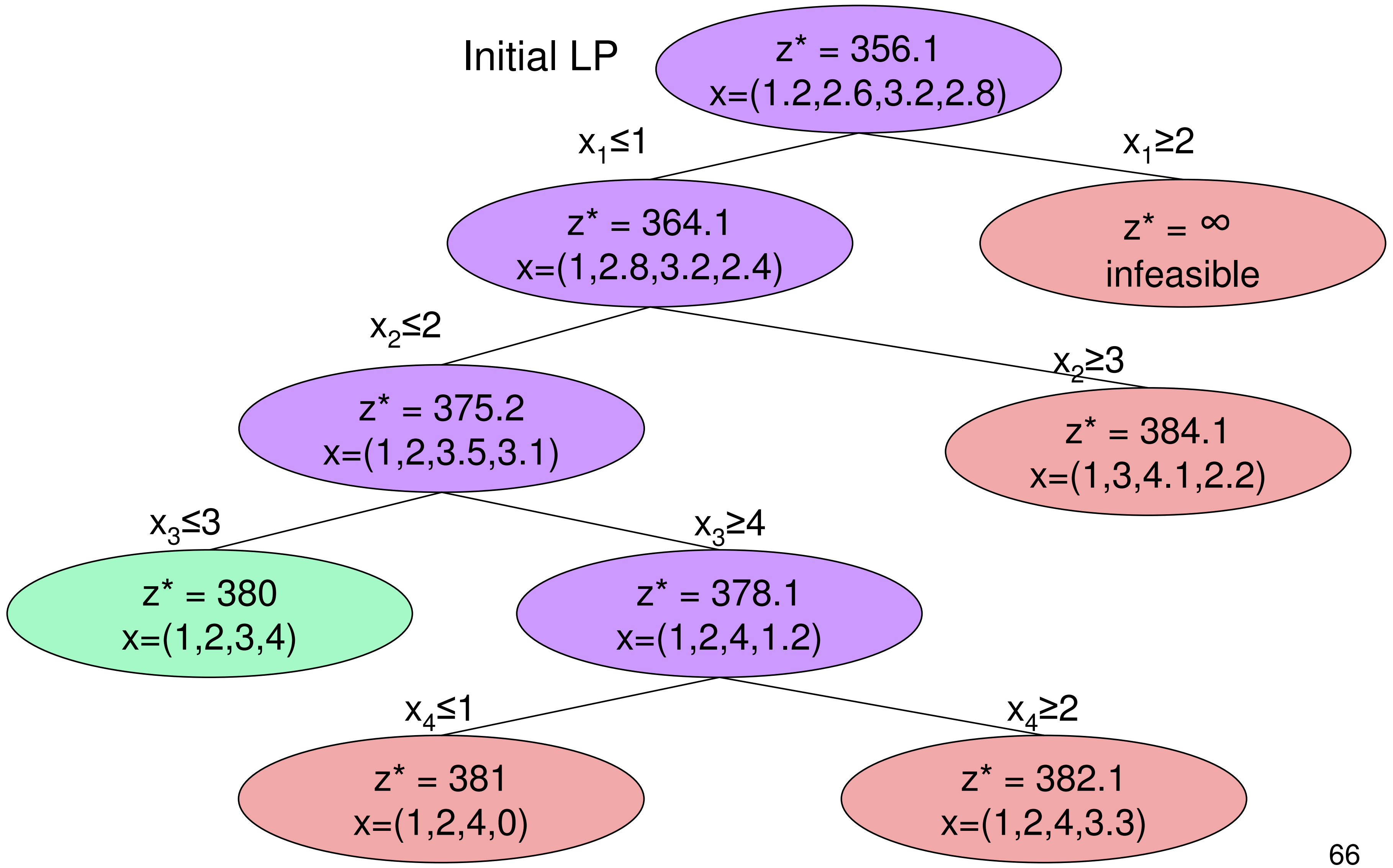
Branch & Bound



Branch & Bound



Branch & Bound



Branch and Bound

- Each integer feasible solution is an upper bound on solution cost,
  - Branching stops
  - It can prune other branches
  - Anytime result: can provide optimality bound
- Each LP-feasible solution is a lower bound on the solution cost
  - Branching may stop if LB ≥ UB

Cutting Planes

- Creating a branch is a lot of work
- Therefore Make bounds tight

- Cutting plane: A new constraint that
  - Keeps all integer solutions
  - Forbids the current fractional LP solution

- First suggested by Gomory even before Simplex was invented
  - “Gomory Cut” is a general cutting plane that can be applied to any LP

Cutting Planes

- Example: Knapsack problem
- Lets say we have the fractional solution

$$x_1 = 0.3,\ x_2 = 0.3, \text{ and } x_3 = 0.5$$

- Assume also that items 1, 2, and 3 are large enough that you cannot select all three
- A valid inequality is

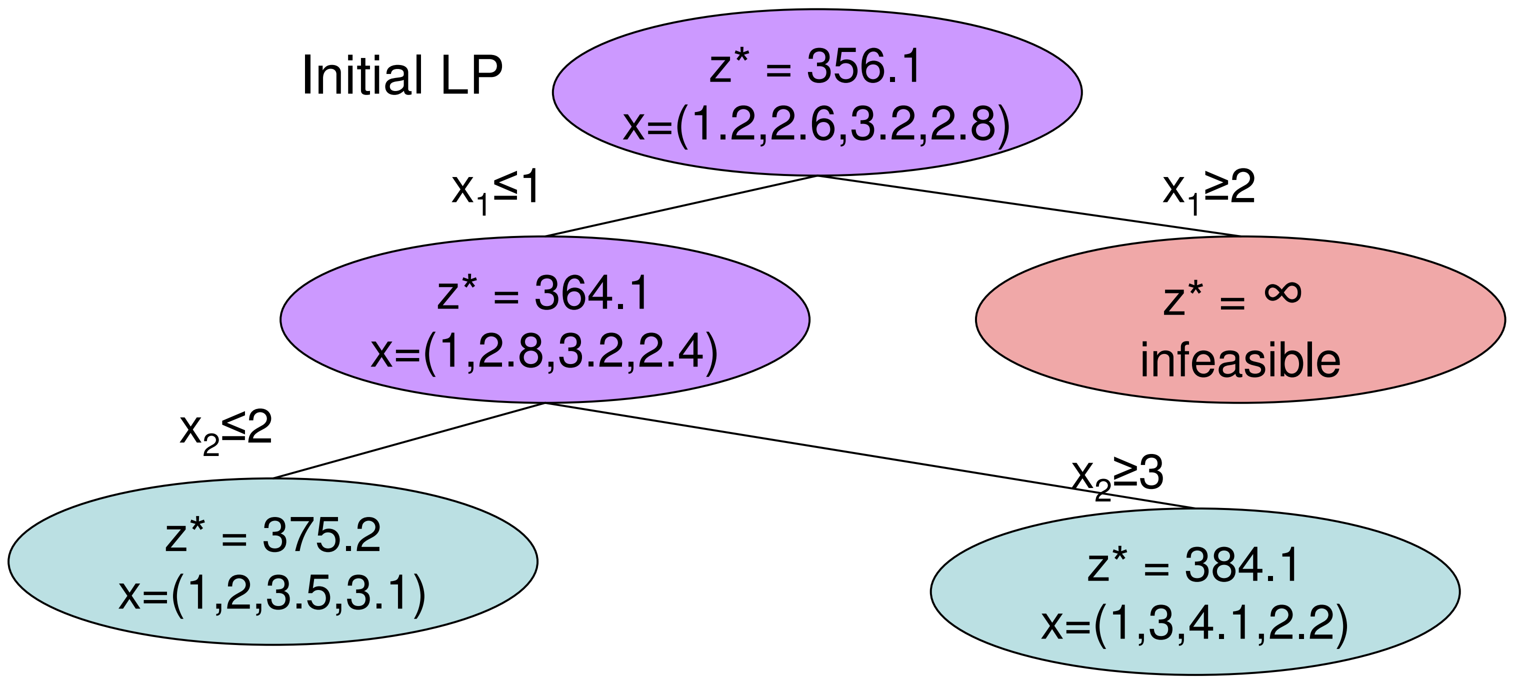
$$x_1 + x_2 + x_3 \leq 1$$

- This forbids the current solution
- … but all legal integer solutions are still valid

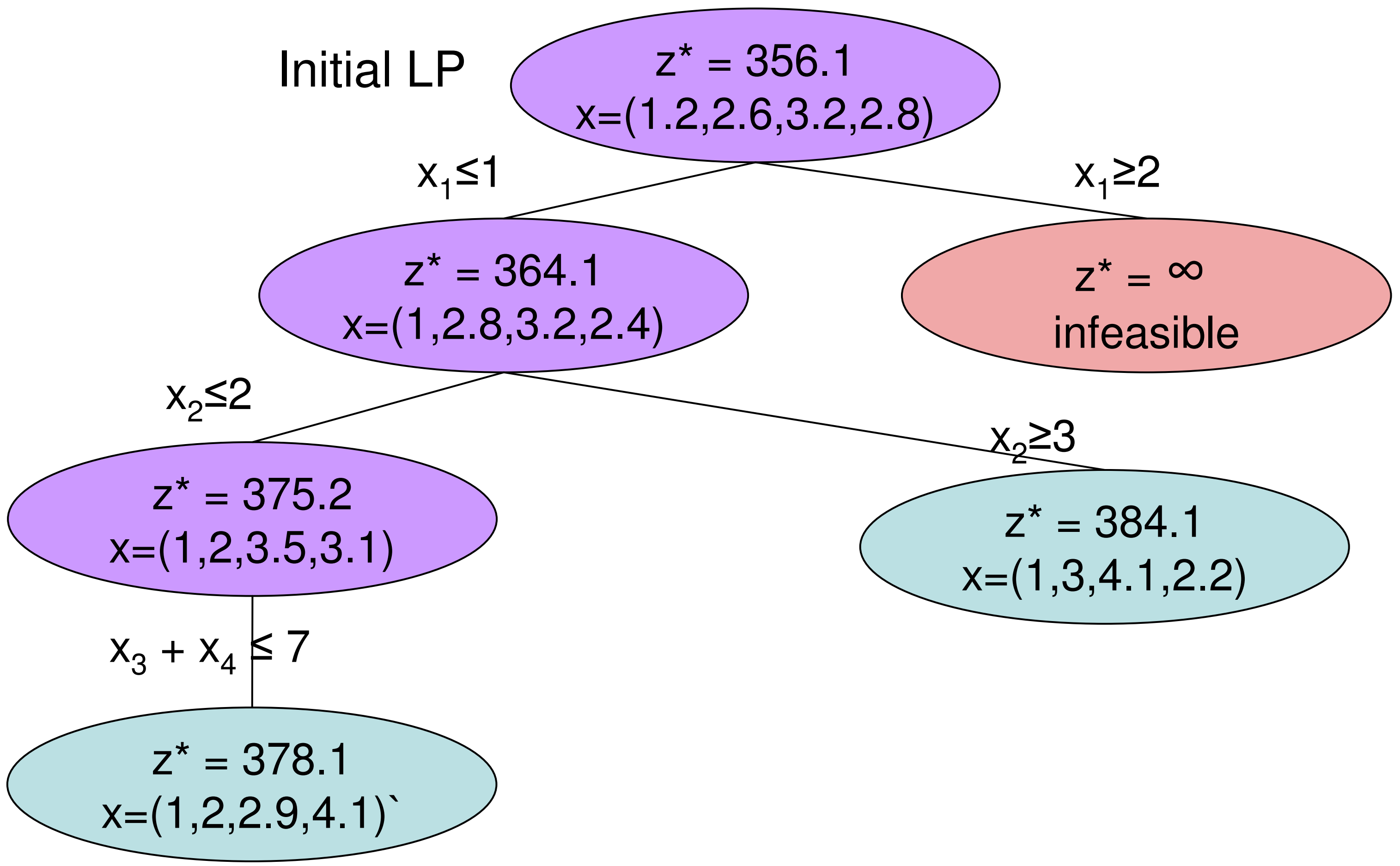
Cutting Planes

- Cutting Planes are applied within a branch-and-bound node to tighten the bound
- Can force a lower-bound high enough that the node is excluded
- May be lucky enough to force an integer solution

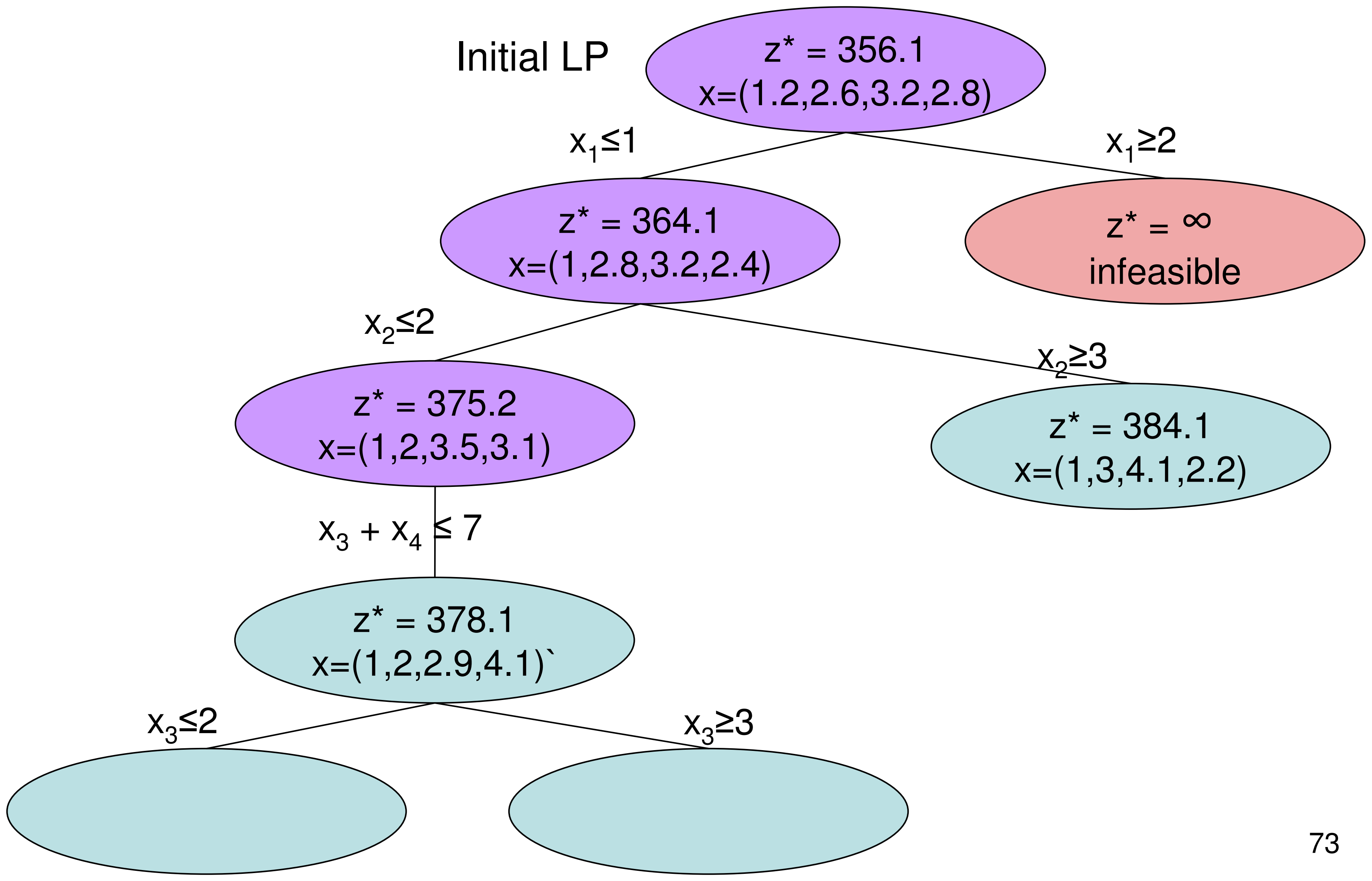
Branch & Bound



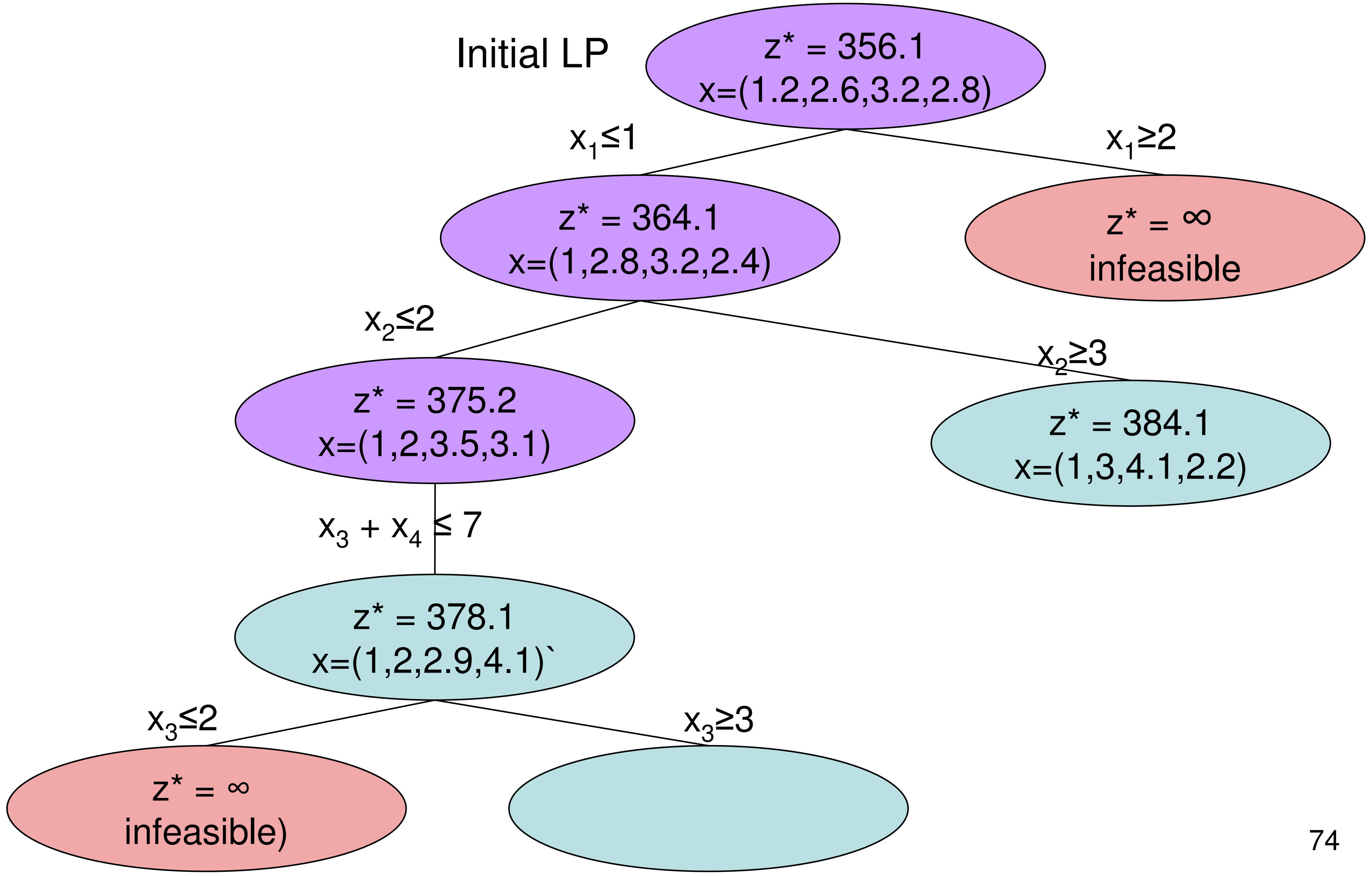
Branch & Bound



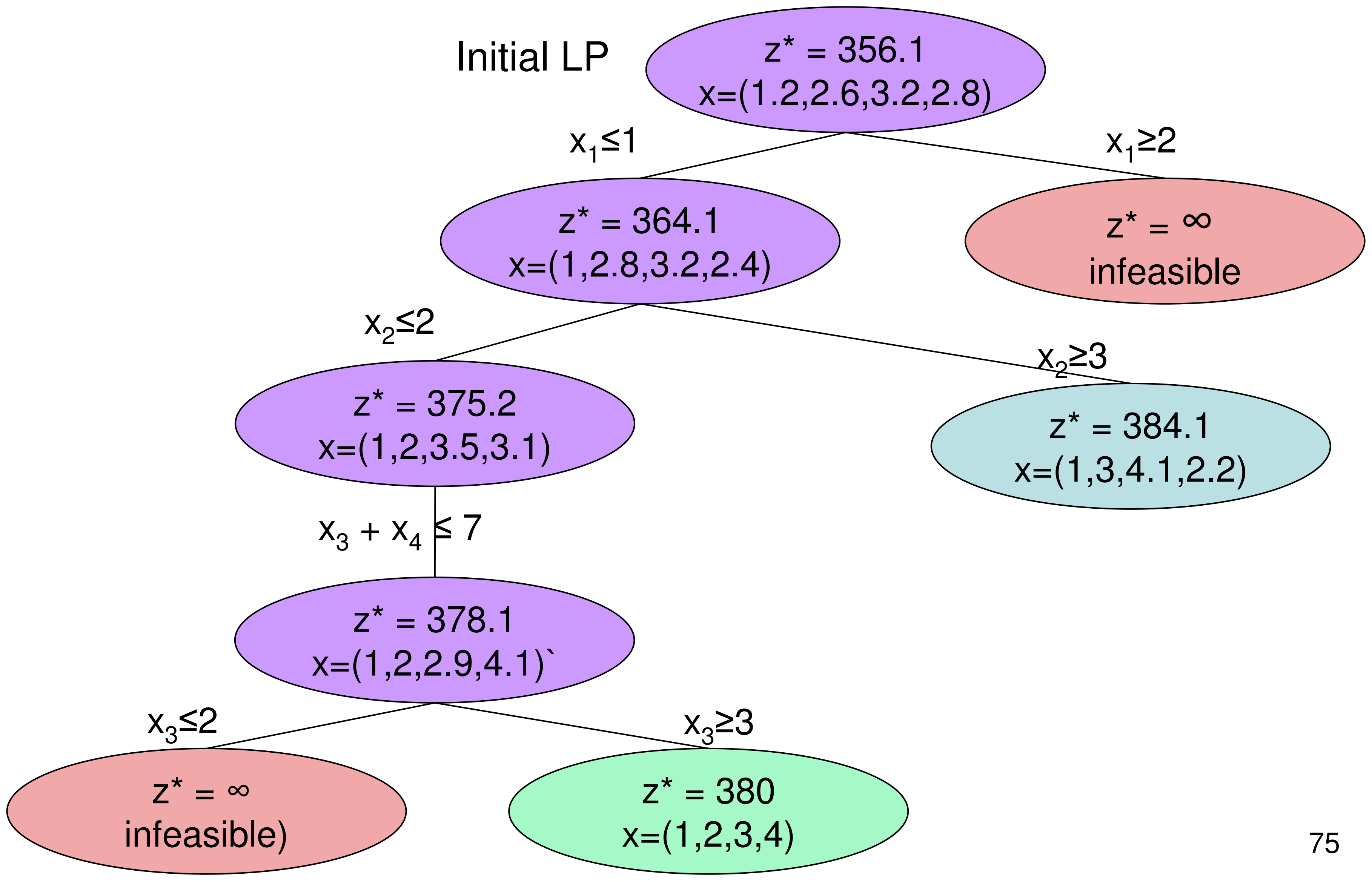
Branch & Bound



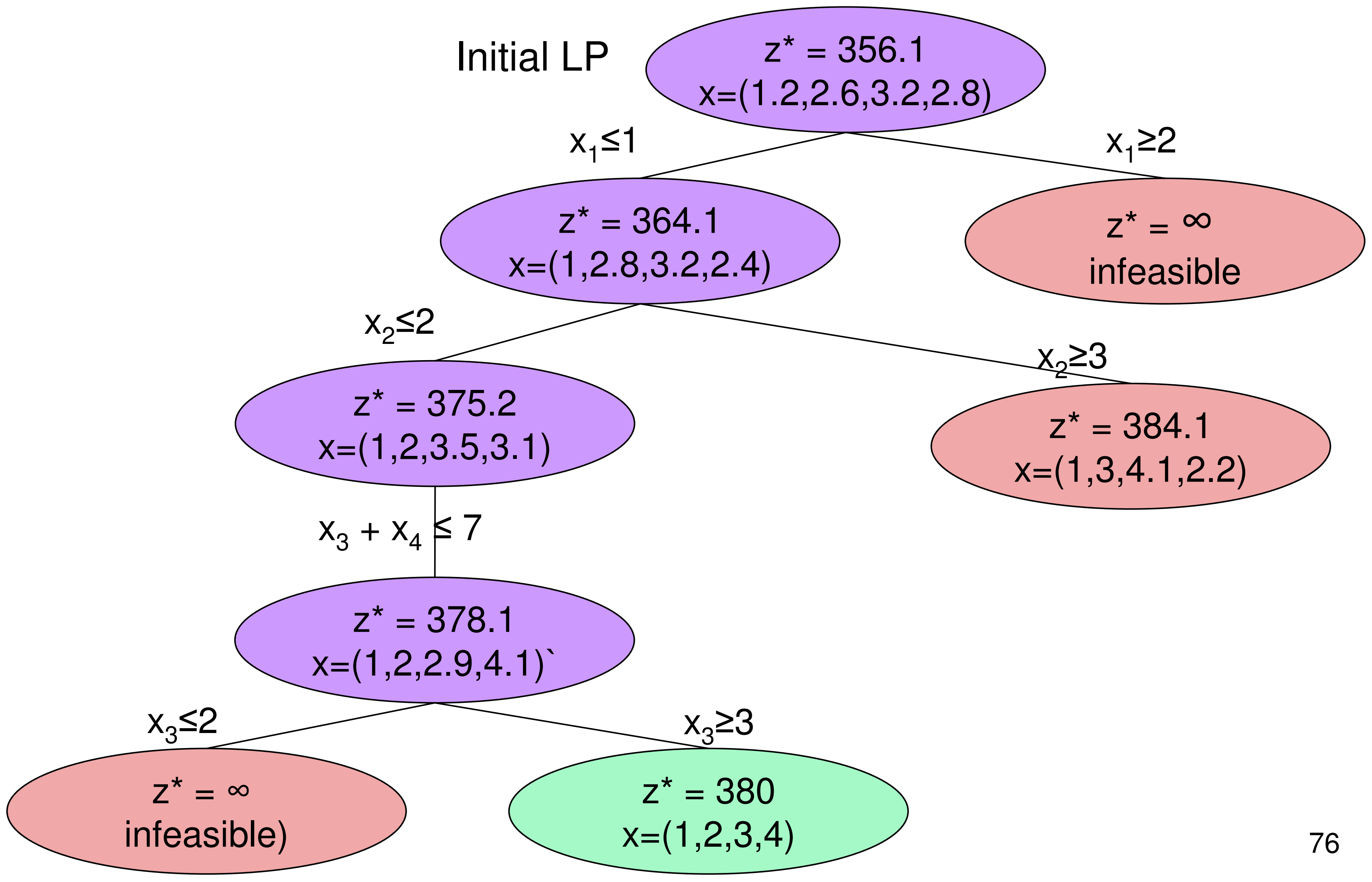
Branch & Bound



Branch & Bound



Branch & Bound



Vehicle Routing Problem

- n customers (n in 100 … 10,000)
- m vehicles
- $c_{i,j}$ – the distance/cost of travel
- q_i – load at customer i
- Q_k – capacity of vehicle k

What vehicle should visit each customer, and in what order, to minimize costs

- 1 vehicle → TSP
- $c_{i,j}$ == 0 → Bin packing

Traditional formulation

- :

$$x_{ijk} = \begin{cases} 1 & \text{if } i \text{ precedes } j \text{ on vehicle } k \\ 0 & \text{otherwise} \end{cases}$$

$$\text{minimize} \sum_{\mathrm{i,j,k}} \mathrm{c_{ij}} x_{ijk}$$

$$\sum_{j} \sum_{k} x_{ijk} = 1$$

$$\sum_{j} x_{ijk} = \sum_{i} x_{ijk} \quad \forall k$$

$$\sum_{i} \sum_{j} x_{ijk} q_j \le Q_k \quad \forall k$$

…

Set Partitioning Formulation

- Create “potential tours” (tour for a single vehicle)
- Save order tour visit customers separately
- Find cost c_j of tour j by solving the associated TSP

Set Partitioning Forumaltion

$$x_i = \begin{cases} 1 & \text{column } i \text{ is used in the solution} \\ 0 & \text{otherwise} \end{cases}$$

$$\min cx$$

$$\sum_j x_j = k \qquad (1)$$

$$\sum_j a_{ij} = 1 \quad \forall i \qquad (2)$$

$$x \in \{0,1\}$$

Set Covering: Replace “=“ in constraint 2 by “≥”

Set Partitioning Formulation

Method:

- Generate a set of columns
- Find cost of each column
- Use Set Partitioning to choose the best set of columns (integer solution required – rats)

But

- Exponential number of possible columns

Column Generation

- Given a solution to the LP, shadow price (reduced cost) r_i of each constraint 2 gives the "value" of each customer at the current solution.
- A column j is guaranteed to enter if

$$\sum_i a_{ij} r_i + c_j < 0$$

Column Generation

- Subproblem is “Constrained, Prize-Collecting Shortest Path”
- Routes must honour all constraints of original problem (e.g. capacity constraints)
- Unfortunately also NP complete
- But good heuristic available

Column Generation

New Method:

- Generate initial columns
- Repeat:
  - Solve [integer] Set Partitioning Problem
  - Generate –ve reduce-cost column(s
- Until no more columns can be produced

- Solution is optimal if method is completed

Next week:

Neighbourhood-based Local Search

Lecture notes available at:

http://users.rsise.anu.edu.au/~pjk/teaching

Task Allocation

- *n* jobs, *m* machines
- Job *i* requires q_i capacity
- At most Q_j assigned to each machine

Forumlation

$$x_{ij} = \begin{cases} 1 & \text{if task } i \text{ is assigned to machine } j \\ 0 & \text{otherwise} \end{cases}$$

$$\min \sum_{i,j} c_{ij} x_{ij}$$

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

Easy subproblem

$$\sum_{i} x_{ij} q_i \le Q_j \quad \forall j$$

Complicating constraints

Lagrangean Relaxation

Problem P:

$$\min \sum_{i,j} c_{ij} x_{ij}$$

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

$$\sum_{i} x_{ij} q_i \le Q_j \quad \forall j$$

Optimum value = z*

Lagrangean Relaxation

$$\min \sum_{i,j} c_{ij} x_{ij}$$

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

$$\sum_{i} x_{ij} q_i - Q_j \quad \forall j$$

-ve: OK
+ve: Amount of infeasibility

Lagrangean Relaxation

$$\min \sum_{i,j} c_{ij} x_{ij}$$

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

$$\sum_{j} \sum_{i} x_{ij} q_i - Q_j$$ Total infeasibility

Lagrangean Relaxation

$$\min \sum_{i,j} c_{ij} x_{ij}$$

$$\sum_j x_{ij} = 1 \quad \forall i$$

$$\sum_j \sum_i x_{ij} q_i - Q_j$$

Total infeasibility

Lagrangean Relaxation

$$\min \sum_{i,j} c_{ij} x_{ij} \qquad + \sum_j \lambda_j \left(\sum_i x_{ij} q_i - Q_j\right)$$

$$\sum_j x_{ij} = 1 \quad \forall i$$

Total infeasibility

Lagrangean Relaxation

$$D(\lambda) = \min_x \sum_{i,j} c_{ij} x_{ij} \qquad + \sum_j \lambda_j \Big(\sum_i x_{ij} q_i - Q_j \Big)$$

$$\sum_j x_{ij} = 1 \quad \forall i$$

Lagrangean Relaxation

- Duality theory tells us that

$$\max_{\lambda \geq 0} D(\lambda) = z^*$$

- and the optimum *x* is the same for both
- (for equality constraints, λ is unconstrained)

- So now we have a continuous optimization problem

Lagrangean Optimization

- Finding

$$\max_{\lambda \geq 0} D(\lambda)$$

can be done via a number of optimization methods….

Next week:

Neighbourhood-based Local Search

Lecture notes available at:

http://users.rsise.anu.edu.au/~pjk/teaching