EL736 Communications Networks II: Design and Algorithms

Class5: Optimization Methods

Yong Liu

10/10/2007

Optimization Methods for NDP

- linear programming
- integer/mixed integer programming
  - NP-Completeness
  - Branch-Bound

Optimization Methods

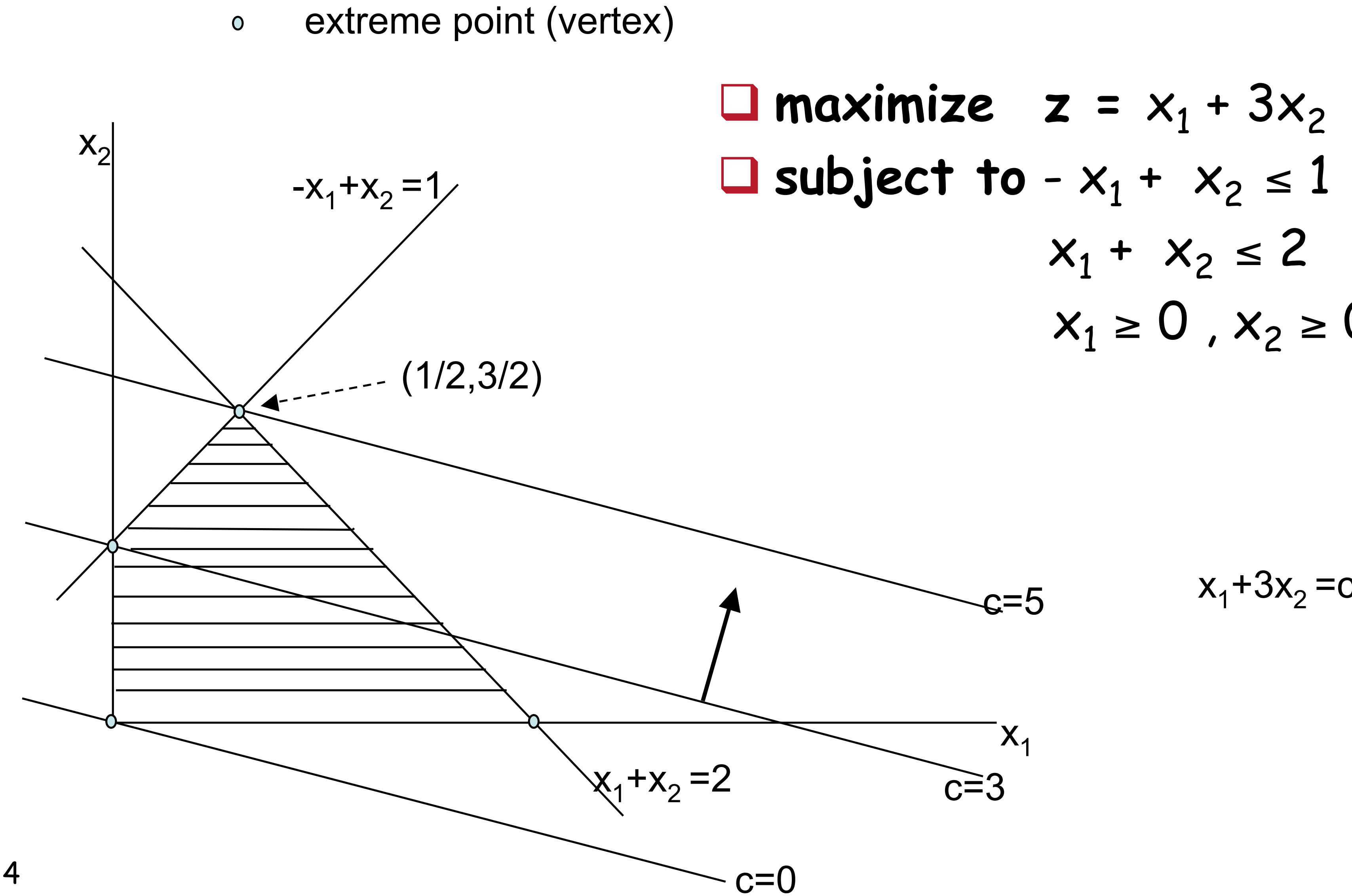
- optimization -- choose the "best".
- what "best" means -- objective function
- what choices you have -- feasible set

$$\min_{X} f(X)$$

$$\text{subject to } X \in A$$

- solution methods
  - brute-force, analytical and heuristic solutions
  - linear/integer/convex programming

Linear Programming - a problem and its solution



- **maximize** $z = x_1 + 3x_2$
- **subject to** $-x_1 + x_2 \leq 1$

  $x_1 + x_2 \leq 2$

  $x_1 \geq 0, x_2 \geq 0$

Linear Program in Standard Form

SIMPLEX

- **indices**
  - $j=1,2,\ldots,n$ — variables
  - $i=1,2,\ldots,m$ — equality constraints
- **constants**
  - $c = (c_1,c_2,\ldots,c_n)$ — cost coefficients
  - $b = (b_1,b_2,\ldots,b_m)$ — constraint left-hand-sides
  - $A = (a_{ij})$ — m × n matrix of constraint coefficients
- **variables**
  - $x = (x_1, x_2,\ldots,x_n)$

Linear program

- maximize

  $z = \Sigma_{j=1,2,\ldots,n}\, c_j x_j$
- subject to

  $\Sigma_{j=1,2,\ldots,m}\, a_{ij} x_j = b_i$, $i=1,2,\ldots,m$

  $x_j \geq 0$, $j=1,2,\ldots,n$

$n > m$

rank($\mathbf{A}$) = m

Linear program (matrix form)

- maximize

  $\mathbf{cx}$
- subject to

  $\mathbf{Ax} = \mathbf{b}$

  $\mathbf{x} \geq \mathbf{0}$

Transformation of LPs to the standard form

- **slack variables**
  - $\Sigma_{j=1,2,\ldots,m} a_{ij}x_j \le b_i$ to $\Sigma_{j=1,2,\ldots,m} a_{ij}x_j + x_{n+i} = b_i$, $x_{n+i} \ge 0$
  - $\Sigma_{j=1,2,\ldots,m} a_{ij}x_j \ge b_i$ to $\Sigma_{j=1,2,\ldots,m} a_{ij}x_j - x_{n+i} = b_i$, $x_{n+i} \ge 0$

- **nonnegative variables**
  - x_k with unconstrained sign: $x_k = x_k' - x_k''$, $x_k' \ge 0$, $x_k'' \ge 0$

<u>Exercise</u>: transform the following LP to the standard form

- **maximize** $z = x_1 + x_2$
- **subject to** $2x_1 + 3x_2 \le 6$

$$x_1 + 7x_2 \ge 4$$
$$x_1 + x_2 = 3$$
$$x_1 \ge 0 , \; x_2 \text{ unconstrained in sign}$$

Basic facts of Linear Programming

- **feasible solution** - satisfying constraints
- **basis matrix** - a non-singular m × m sub-matrix of *A*
- **basic solution** to a LP - the unique vector determined by a basis matrix: n-m variables associated with columns of *A* not in the basis matrix are set to 0, and the remaining m variables result from the square system of equations
- **basic feasible solution** - basic solution with all variables nonnegative (at most m variables can be positive)
- **extreme point** - feasible point cannot be expressed as a convex linear combination of other feasible points

$$x \neq \sum_{k=1}^{K} \alpha_k x_k, \quad (\alpha_k \geq 0, \sum_{k=1}^{K} \alpha_k = 1)$$

Basic facts of Linear Programming

- **Theorem 1.**

  The objective function, z, assumes its maximum at an extreme point of the constraint set.

- **Theorem 2.**

  A vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is an extreme point of the constraint set if and only if $\mathbf{x}$ is a basic feasible solution.

Capacitated flow allocation problem - LP formulation

- **variables**
  - x_{dp} flow realizing demand d on path p

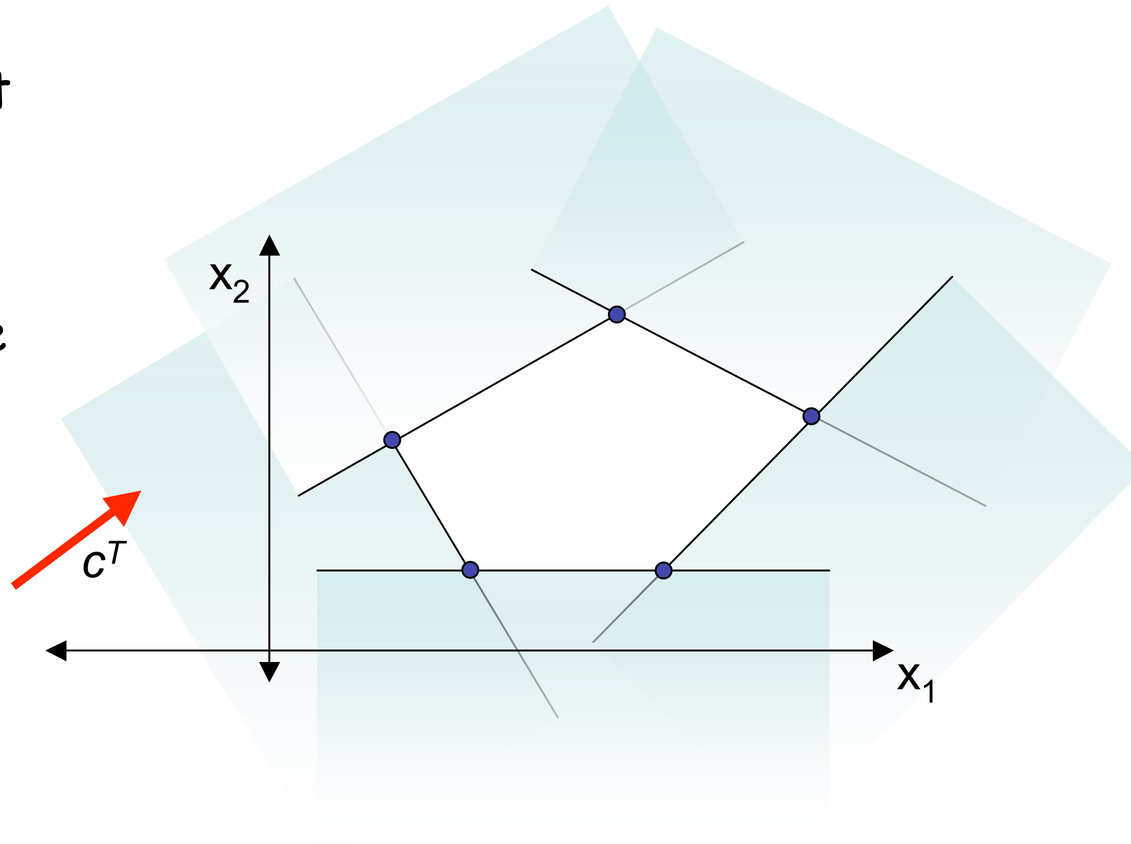
- **constraints**
  - $\Sigma_p \, x_{dp} = h_d$ d=1,2,…,D
  - $\Sigma_d \, \Sigma_p \, \delta_{edp} x_{dp} \leq c_e$ e=1,2,…,E
  - flow variables are *continuous and non-negative*

- **Property:**

  D+E non-zero flows at most
  - depending on the number of saturated links
  - if all links unsaturated: D flows only!

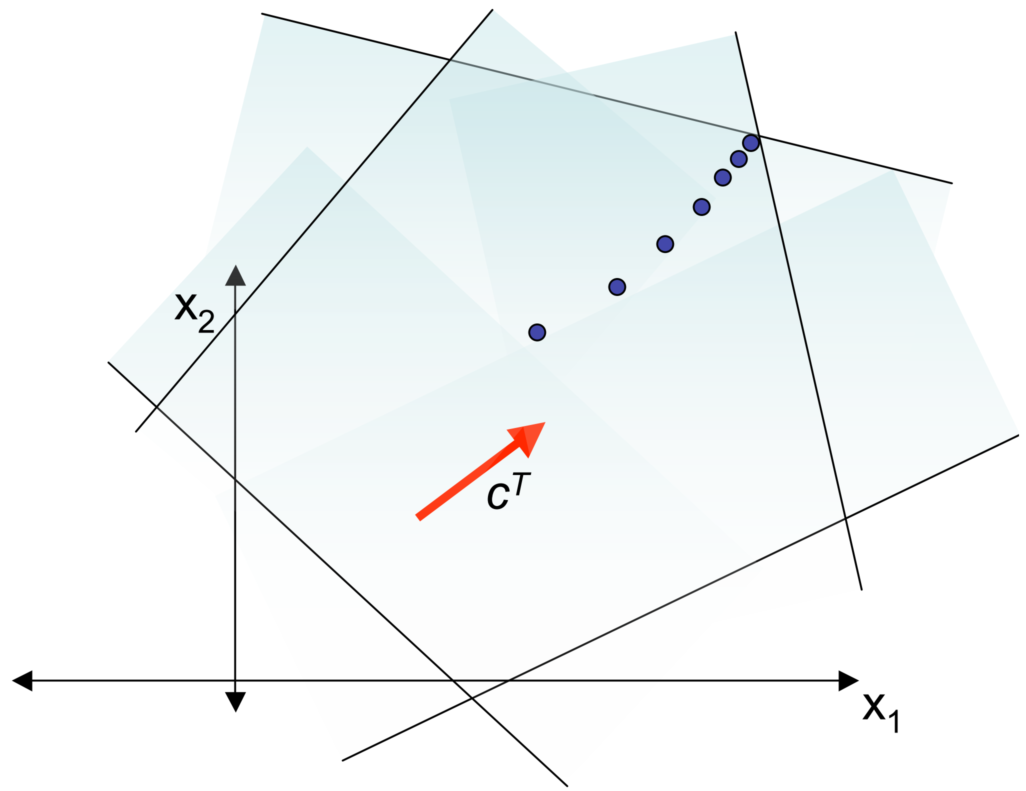
Solution Methods for Linear Programs

- Simplex Method
  - Optimum must be at the intersection of constraints
  - Intersections are easy to find, change inequalities to equalities
  - Jump from one vertex to another
  - Efficient solution for most problems, exponential time worst case.



Solution Method for Linear Programs

- Interior Point Methods
  - Apply Barrier Function to each constraint and sum
  - Primal-Dual Formulation
  - Newton Step
- Benefits
  - Scales Better than Simplex
  - Certificate of Optimality
  - Polynomial time algorithm



IPs and MIPs

Integer Program (IP)

maximize	$z = \mathbf{c}\mathbf{x}$	
subject to	$\mathbf{A}\mathbf{x} \leq \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$	(linear constraints)
	$\mathbf{x}$ integer	(integrality constraint)

Mixed Integer Program (MIP)

maximize	$z = \mathbf{c}\mathbf{x} + \mathbf{d}\mathbf{y}$	
subject to	$\mathbf{A}\mathbf{x} + \mathbf{D}\mathbf{y} \leq \mathbf{b}$, $\mathbf{x}, \mathbf{y} \geq \mathbf{0}$	(linear constraints)
	$\mathbf{x}$ integer	(integrality constraint)

Complexity: NP-Complete Problems

- Problem Size n: variables, constraints, value bounds.
- Time Complexity: asymptotics when n large.
  - polynomial: n^k
  - exponential: k^n
- The *NP-Complete* problems are an interesting class of problems whose status is unknown
  - no polynomial-time algorithm has been discovered for an NP-Complete problem
  - no supra-polynomial lower bound has been proved for any NP-Complete problem, either
  - All NP-Complete problems "equivalent".

Prove NP-Completeness

- Why?
  - most people accept that it is probably intractable
  - don't need to come up with an efficient algorithm
  - can instead work on *approximation algorithms*
- How?
  - reduce (transform) a well-known NP-Complete problem P into your own problem Q
  - if P reduces to Q, P is "no harder to solve" than Q

IP (and MIP) is NP-Complete

- SATISFIABILTY PROBLEM (SAT) can be expressed as IP
- even as a binary program (all integer variables are binary)

SATISFIABILITY PROBLEM

SAT

$U = \{u_1, u_2, \ldots u_m\}$ - Boolean variables; $t : U \rightarrow \{\underline{true}, \underline{false}\}$ - truth assignment

a clause - $\{u_1, \underline{u}_2, u_4\}$ represents conjunction of its elements $(u_1 + \underline{u}_2 + u_4)$

a clause is satisfied by a truth assignment t if and only if one of its elements is true under assignment t

C - finite collection of n clauses

SAT:	given:	a set U of variables and a collection C of clauses
	question:	is there a truth assignment satisfying all clauses in C?

All problems in class NP can be reduced to SAT (Cook's theorem)

So far there are several thousands of known NP problems, (including Travelling Salesman, Clique, Steiner Problem, Graph Colourability, Knapsack) to which SAT can be reduced

Integer Programming is NP-Complete

X - set of vectors $\mathbf{x} = (x_1,x_2,\ldots,x_n)$
$\mathbf{x} \in X$ iff $\mathbf{Ax} \leq \mathbf{b}$ and $\mathbf{x}$ are integers

Decision problem:

Instance: given n, **A**, **b**, *C*, and linear function f(**x**).

Question: is there $\mathbf{x} \in X$ such that $f(\mathbf{x}) \leq C$?

The SAT problem is directly reducible to a binary IP problem.

- assign binary variables x_i and $\underline{x}_i$ with each Boolean variables u_i and $\underline{u}_i$
- an inequality for each clause of the instance of SAT ($x_1 + \underline{x}_2 + x_4 \geq 1$)
- add inequalities: $0 \leq x_i \leq 1$, $0 \leq \underline{x}_i \leq 1$, $1 \leq x_i + \underline{x}_i \leq 1$, $i=1,2,\ldots,n$

Optimization Methods for MIP and IP

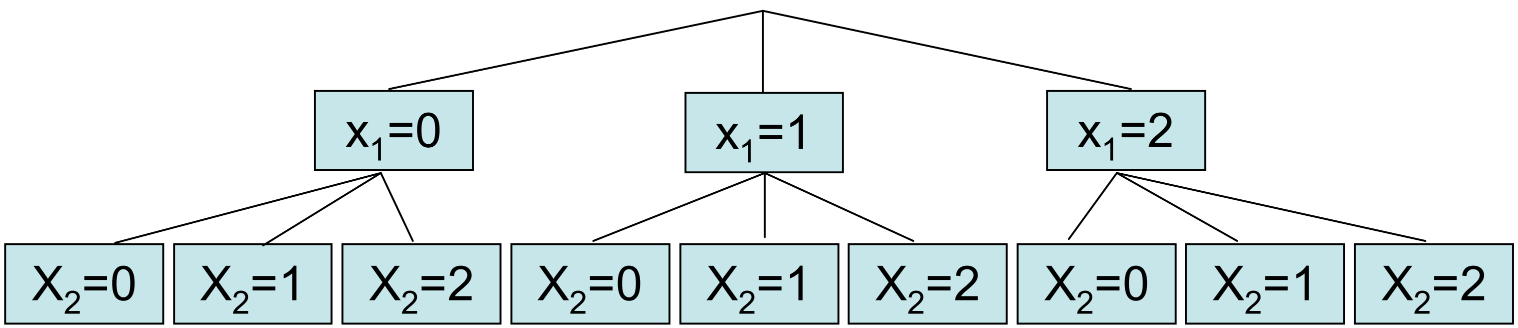
- no hope for efficient (polynomial time) exact general methods
- main stream for achieving exact solutions:
  - branch-and-bound
  - branch-and-cut
  - based on using LP
  - can be enhanced with Lagrangean relaxation
- stochastic heuristics
  - evolutionary algorithms, simulated annenaling, etc.

Why LPs, MIPs, and IPs are so Important?

- in practice only LP guarantees efficient solutions
- decomposition methods are available for LPs
- MIPs and IPs can be solved by general solvers by the branch-and-cut method, based on LP
  - CPLEX, XPRESS
  - sometimes very efficiently
- otherwise, we have to use (frequently) unreliable stochastic meta-heuristics (sometimes specialized heuristics)

Solution Methods for Integer Programs

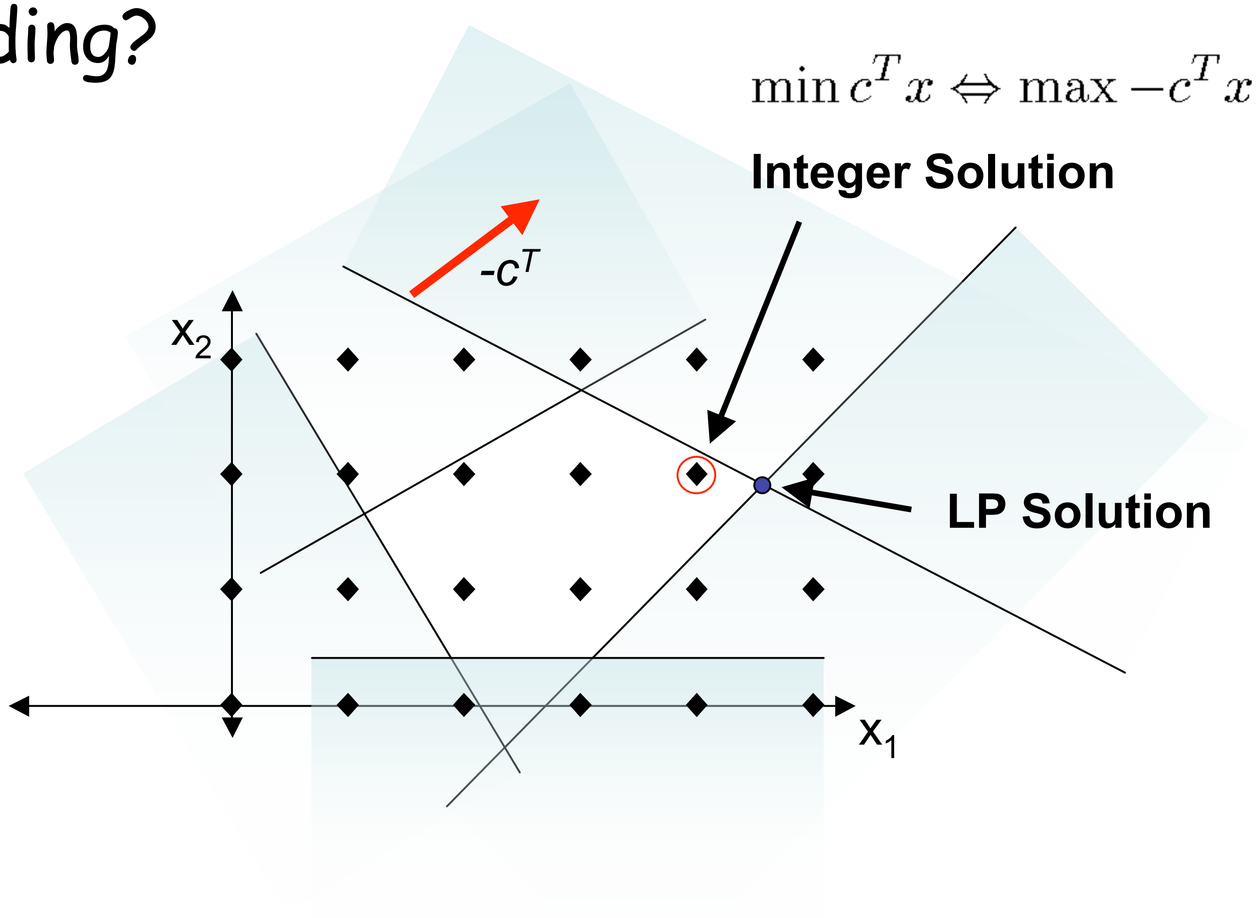
- Enumeration – Tree Search, Dynamic Programming etc.



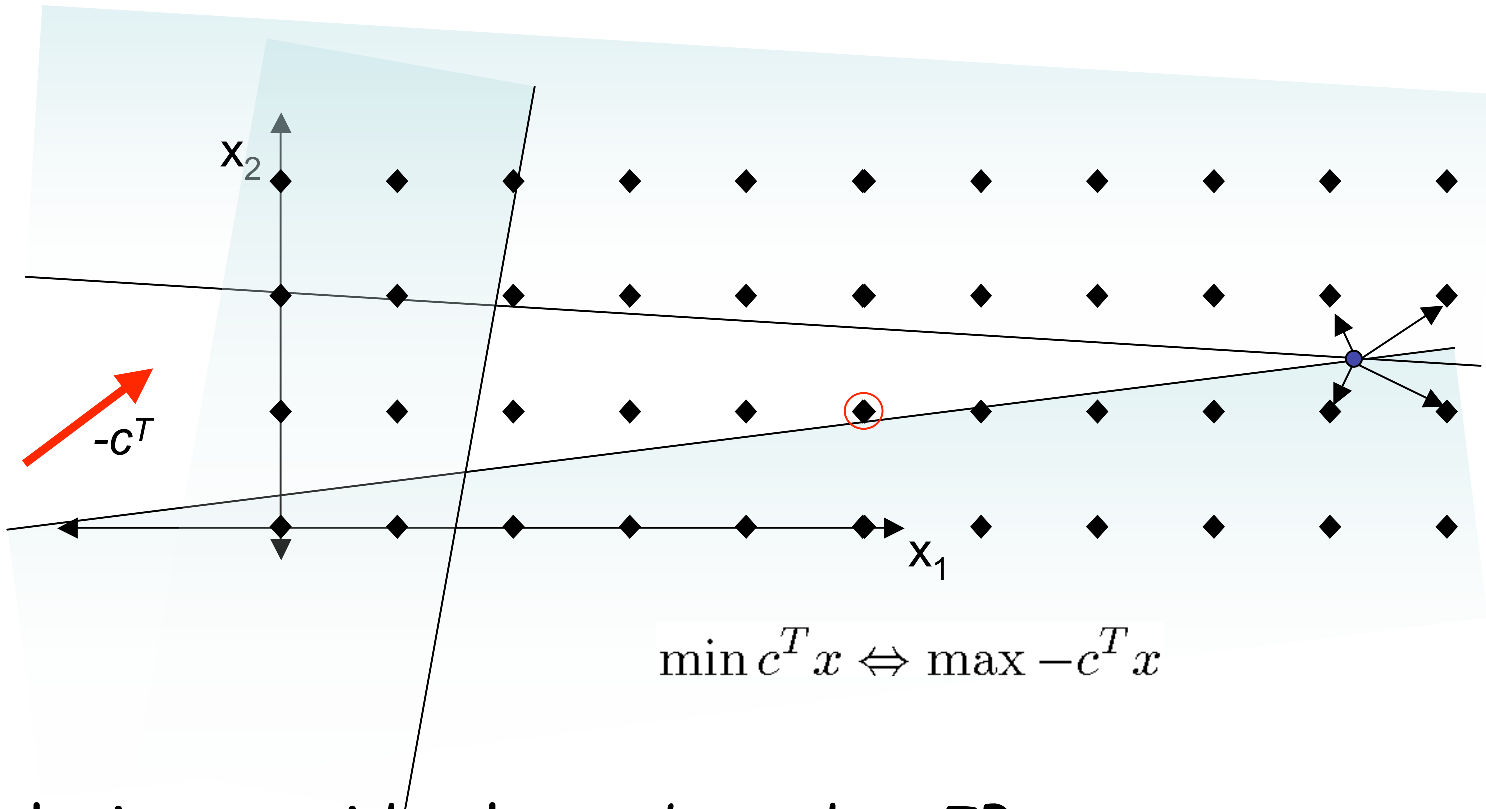
  - Guaranteed to find a feasible solution (only consider integers, can check feasibility (P))
  - But, exponential growth in computation time

Solution Methods for Integer Programs

- How about solving LP Relaxation followed by rounding?



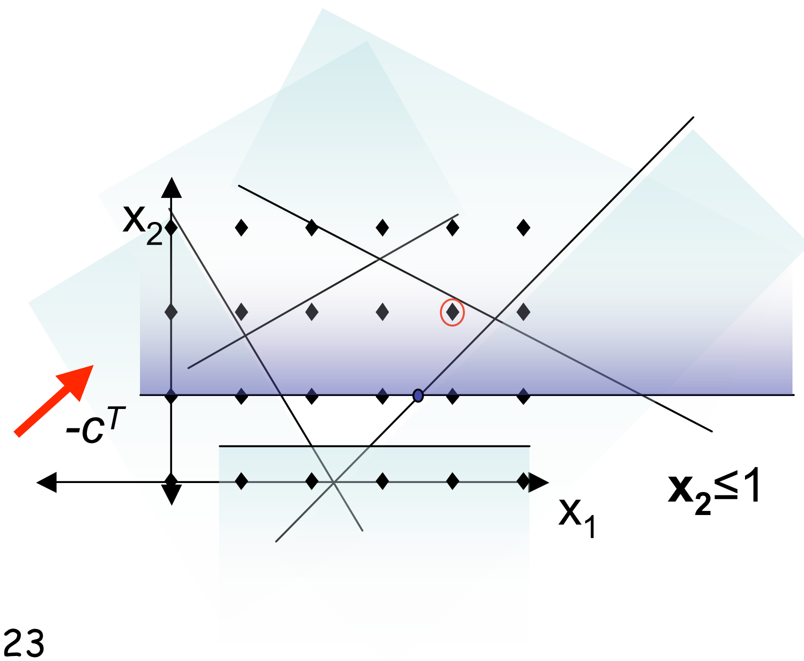
Integer Programs

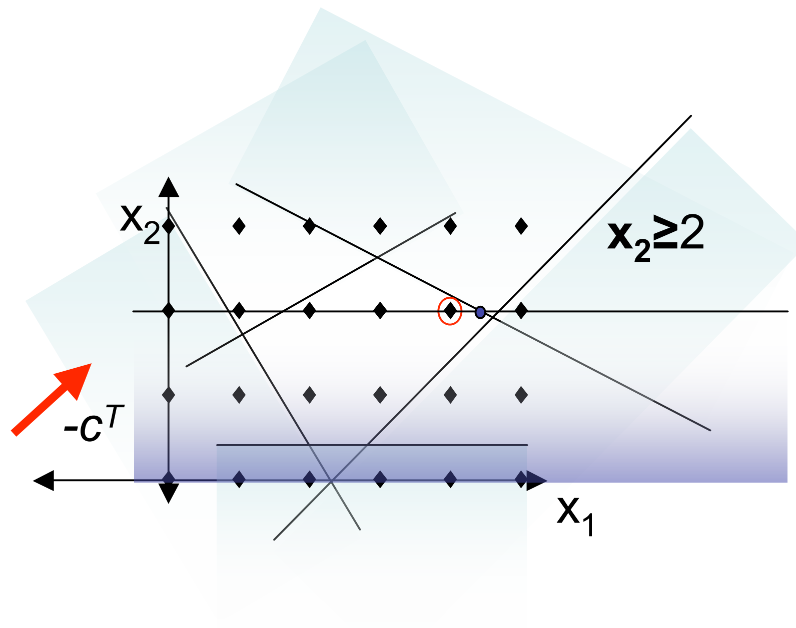


- LP solution provides lower bound on IP
- But, rounding can be arbitrarily far away from integer solution

Combined approach to Integer Programming

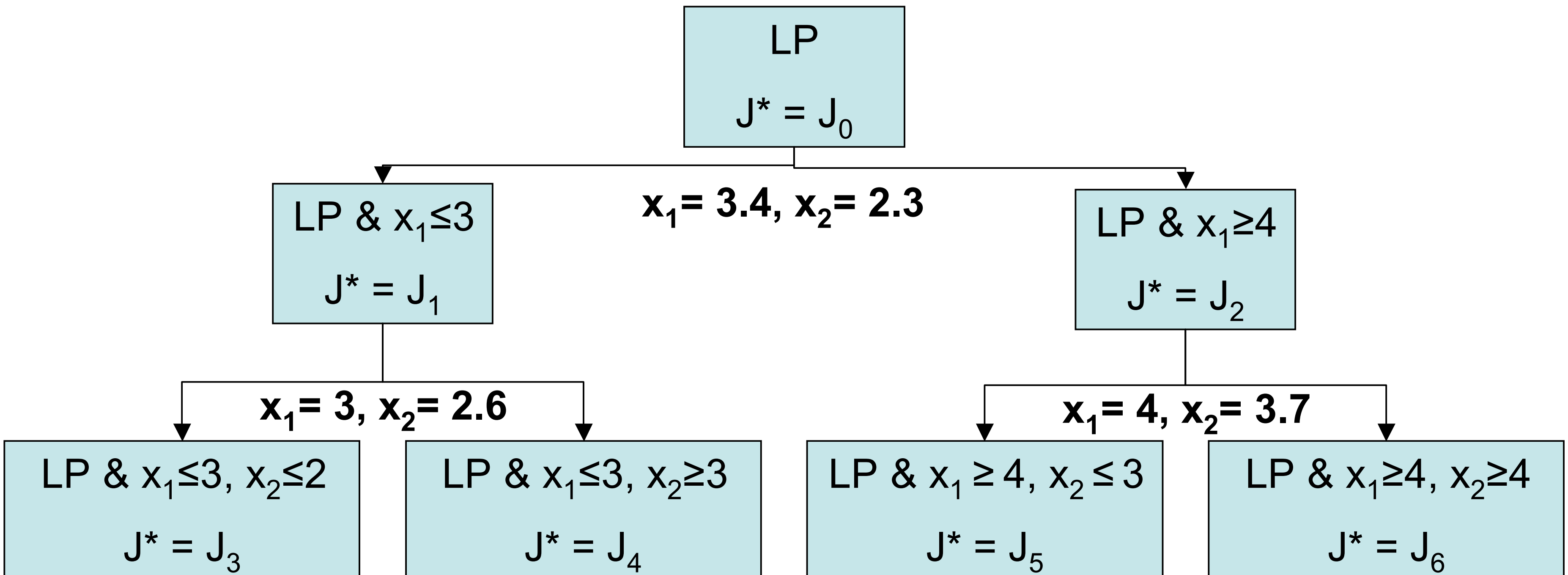
- Why not combine both approaches!
  - Solve LP Relaxation to get fractional solutions
  - Create two sub-branches by adding constraints





Solution Methods for Integer Programs

- Known as Branch and Bound
  - Branch as above
  - For minimizing problem, LP give lower bound, feasible solutions give upper bound

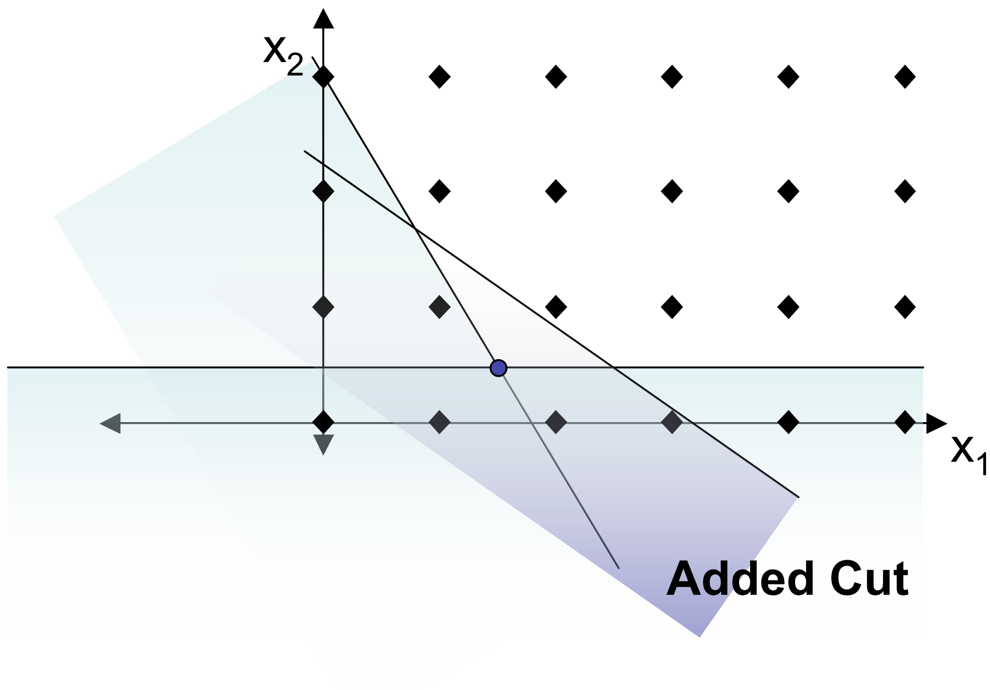


Branch and Bound Method for Integer Programs

- 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.

Additional Refinements – Cutting Planes

- Idea stems from adding additional constraints to LP to improve tightness of relaxation
- Combine constraints to eliminate non-integer solutions



- *All feasible integer solutions remain feasible*
- Current LP solution is not feasible

General B&B algorithim for the binary case

- Problem P
  - minimize $z = \mathbf{cx}$
  - subject to $\mathbf{Ax} \leq \mathbf{b}$
    - $x_i \in \{0,1\}$, $i=1,2,\ldots,k$
    - $x_i \geq 0$, $i=k+1,k+2,\ldots,n$

- $N_U, N_0, N_1 \subseteq \{1,2,\ldots,k\}$ partition of $\{1,2,\ldots,k\}$
- $P(N_U,N_0,N_1)$ – relaxed problem in continuous variables x_i, $i \in N_U \cup \{k+1,k+2,\ldots,n\}$
  - $0 \leq x_i \leq 1$, $i \in N_U$
  - $x_i \geq 0$, $i=k+1,k+2,\ldots,n$
  - $x_i = 0$, $i \in N_0$
  - $x_i = 1$, $i \in N_1$
- $z^{best} = +\infty$

B&B for the binary case - algorithm

procedure BBB(N_U,N_0,N_1)

begin

    solution(N_U,N_0,N_1,$\mathbf{x}$,z); { solve P(N_U,N_0,N_1) }

    if $N_U = \emptyset$ **or** for all $i \in N_U$ x_i are binary **then**

        if $z < z^{best}$ **then begin** $z^{best} := z$; $\mathbf{x}^{best} := \mathbf{x}$ **end**

    else

        if $z \geq z^{best}$ **then**

            return { *bounding* }

        else

            begin { *branching* }

             choose $i \in N_U$ such that x_i is fractional;

             BBB($N_U \setminus \{ i \}$,$N_0 \cup \{ i \}$,N_1); BBB($N_U \setminus \{ i \}$,N_0,$N_1 \cup \{ i \}$)

            end

end { procedure }

B&B - example

original problem:

(IP) maximize **cx**

subject to **Ax** ≤ **b**

x ≥ **0** and integer

linear relaxation:

(LR) maximize **cx**

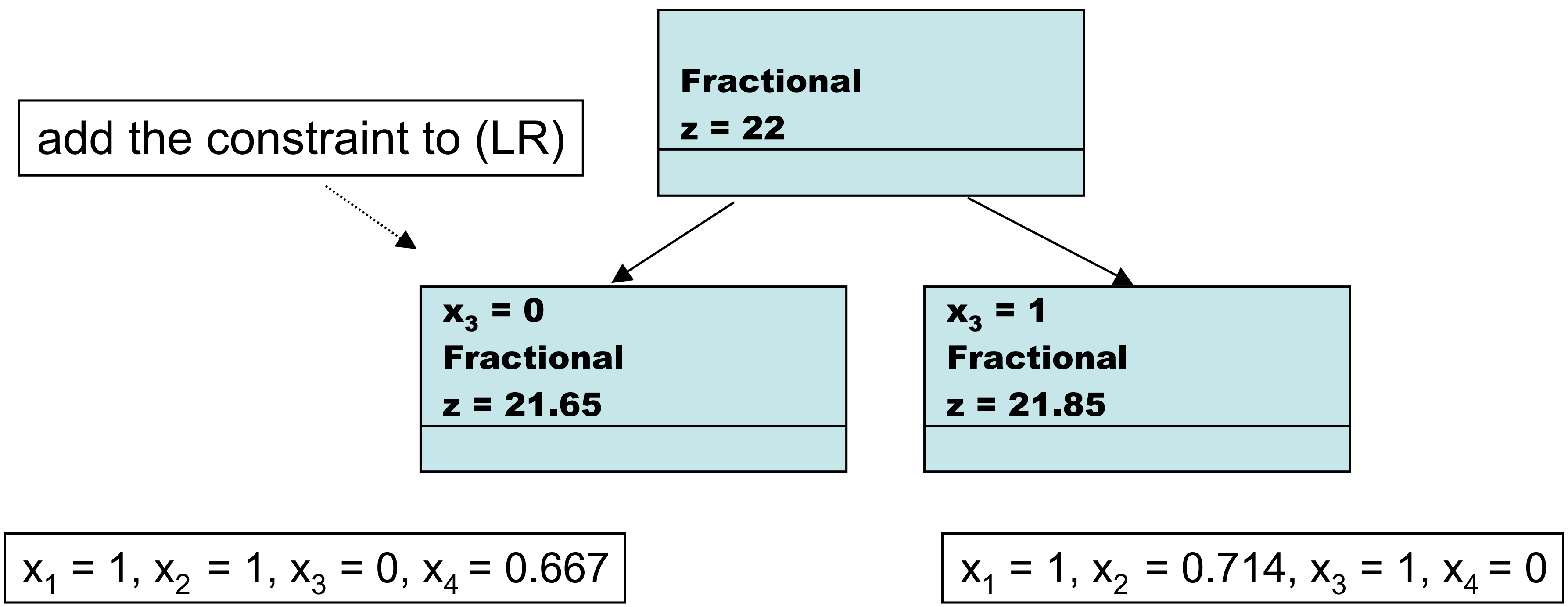
subject to **Ax** ≤ **b**

x ≥ **0**

- The optimal objective value for (LR) is greater than or equal to the optimal objective for (IP).
- If (LR) is infeasible then so is (IP).
- If (LR) is optimised by integer variables, then that solution is feasible and optimal for (IP).
- If the cost coefficients c are integer, then the optimal objective for (IP) is less than or equal to the “round down” of the optimal objective for (LR).

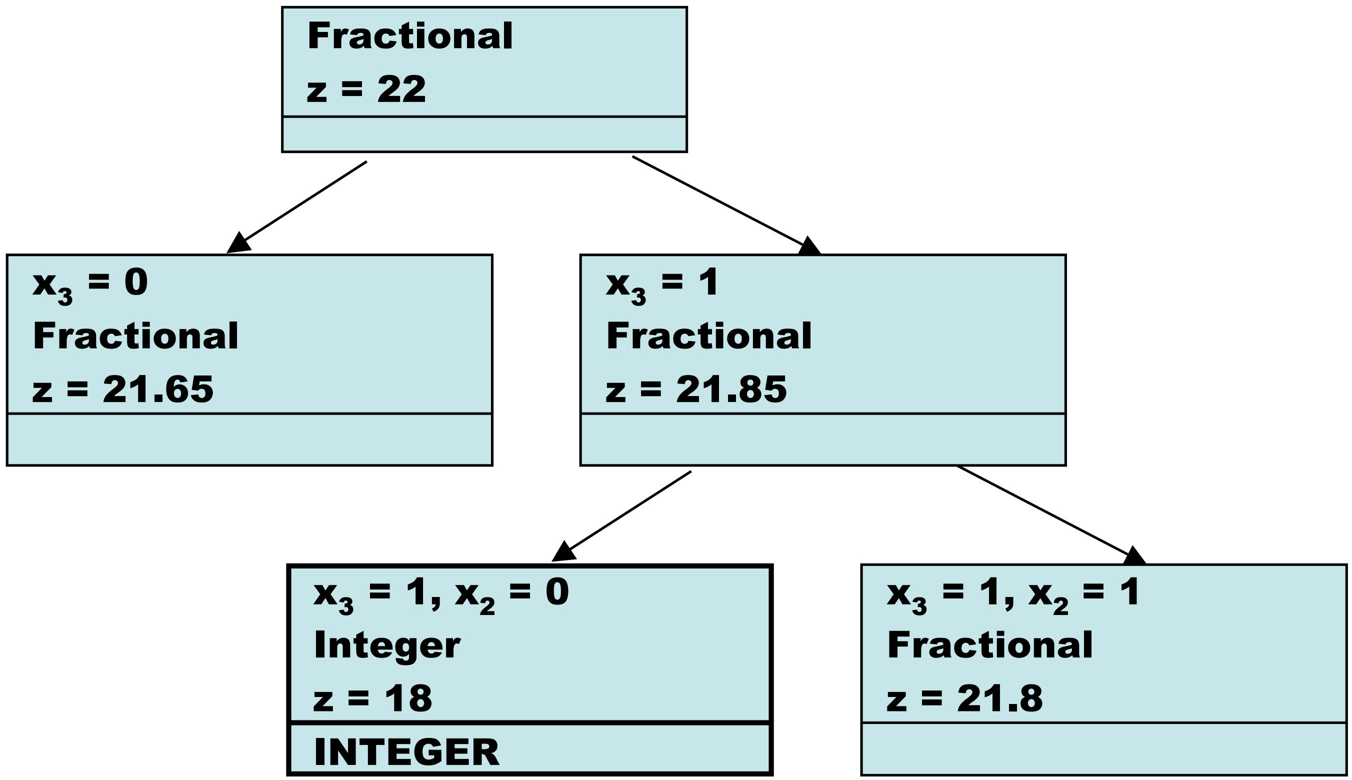
B&B - knapsack problem

- **maximize** $8x_1 + 11x_2 + 6x_3 + 4x_4$
- **subject to** $5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14$
- $x_j \in \{0,1\}$, $j=1,2,3,4$
- (LR) solution: $x_1 = 1$, $x_2 = 1$, $x_3 = 0.5$, $x_4 = 0$, $z = 22$
  - no integer solution will have value greater than 22



B&B example cntd.

- we know that the optimal integer solution is not greater than 21.85 (21 in fact)
- we will take a sub-problem and branch on one of its variables
  - \- we choose an active sub-problem (here: not chosen before)
  - \- we choose a sub-problem with highest solution value

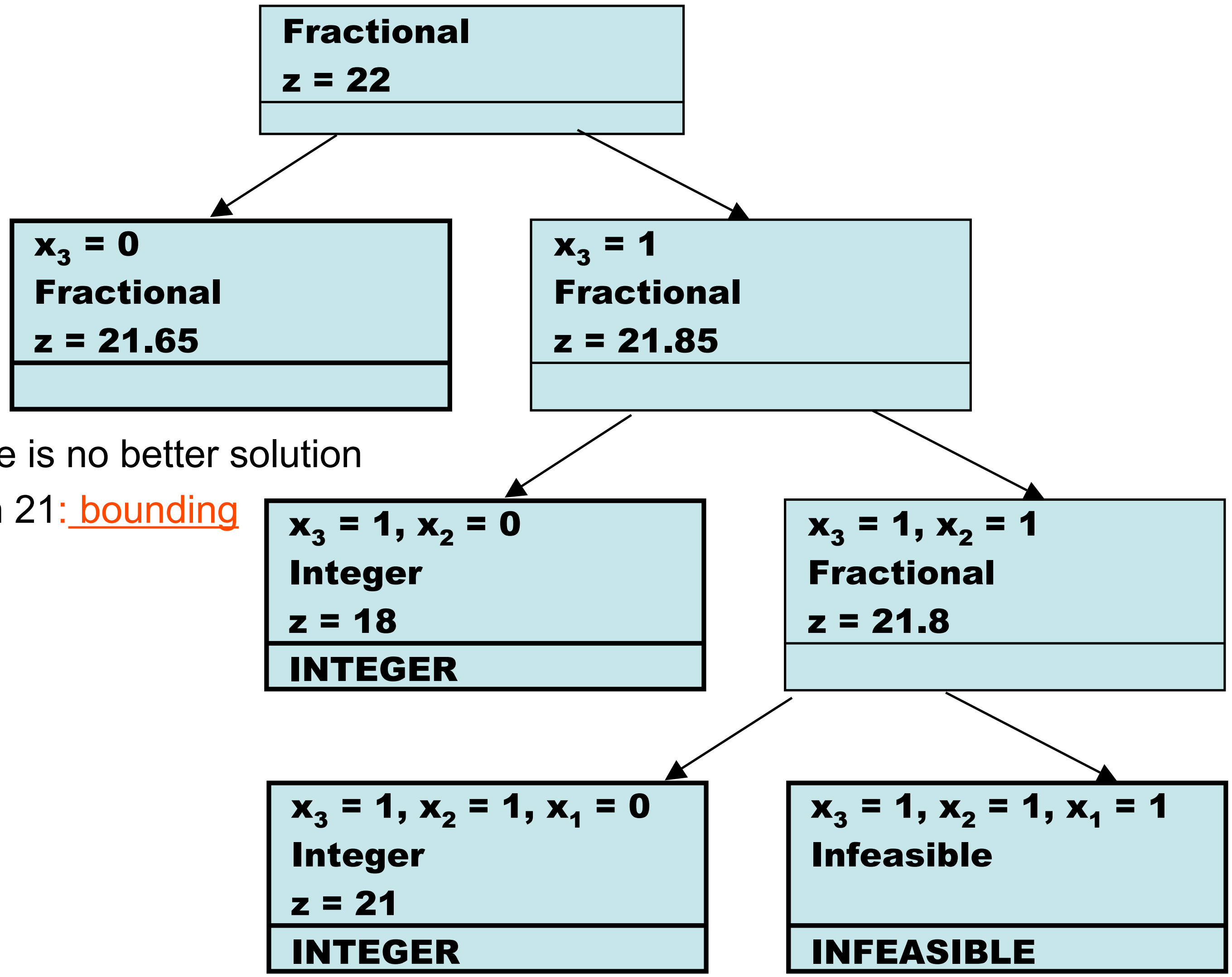


no further branching, not active

$x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 1$

$x_1 = 0.6, x_2 = 1, x_3 = 1, x_4 = 0$

B&B example cntd.



there is no better solution than 21: bounding

$x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 1$ optimal

$x_1 = 1, x_2 = 1, x_3 = 1, x_4 = ?$

B&B example - summary

- Solve the linear relaxation of the problem. If the solution is integer, then we are done. Otherwise create two new sub-problems by branching on a fractional variable.
- A sub-problem is not active when any of the following occurs:
  - you have already used the sub-problem to branch on
  - all variables in the solution are integer
  - the subproblem is infeasible
  - you can bound the sub-problem by a bounding argument.
- Choose an active sub-problem and branch on a fractional variable. Repeat until there are no active sub-problems.
- **Remarks**
  - If x is restricted to integer (but not necessarily to 0 or 1), then if $x = 4.27$ you would branch with the constraints $x \leq 4$ and $x \geq 5$.
  - If some variables are not restricted to integer you do not branch on them.

B&B algorithim - comments

- Also, integer MIP can always be converted into binary MIP

  transformation: $x_j = 2^0 u_{j0} + 2^1 u_{j1} + \ldots + 2^q u_{jq}$ $(x_j \leq 2^{q+1} - 1)$

- Lagrangean relaxation can also be used for finding lower bounds (instead of linear relaxation).
- **Branch-and-Cut (B&C)**
  - combination of B&B with the cutting plane method
  - the most effective exact approach to NP-complete MIPs
  - idea: add "valid inequalities" which define the <u>facets</u> of the integer polyhedron

Next Lecture

- AMPL/CPLEX Package

- Stochastic Methods