Universiteit Utrecht

[Faculty of Science
Information and Computing Sciences]

Linear programming
and
Integer Linear Programming

Based on material by
Marjan van den Akker

Some changes by Hans Bodlaender

In these lectures

- Modelling problems as a linear program or integer linear program
- Solving LPs
- Solving ILPs

An introduction to

LINEAR PROGRAMS

Universiteit Utrecht

[Faculty of Science
Information and Computing Sciences]

Linear Programming

- Linear Programming: Form of problems that
  - Can be used to express many questions from applications
  - Can be solved efficiently
  - Tools available
- Related to Integer Linear Programming
  - "Same thing", except that all values of the solution must be integers
  - Can be used to express many questions from applications
  - Sometimes can be solved efficiently
  - Tools available
- And we also have Mixed Integer Linear Programming
  - Some values of the solution must be integer
  - Can be etc.

Choices for an investor

- Suppose we can now buy and sell later two products, say grain and salt
- We can invest A euro's
- Grain costs B per unit
- Salt costs C per unit
- We can store at most D units
- After a month, we can sell grain for E and salt for F
- We can buy/sell at most G units of grain and H units of salt
- We can buy/sell fractions of units as well

- How can we earn as much money as possible?

Formulate constraints

- Variable x: amount of grain we buy and sell
- Variable y: amount of salt we buy and sell

- Grain costs B and salt costs C; we can invest A

Formulate constraints

- Variable x: amount of grain we buy and sell
- Variable y: amount of salt we buy and sell

- Grain costs B and salt costs C; we can invest A
  - $Bx + Cy \leq A$
- We can store at most D units
  - $x + y \leq D$

Formulate constraints

- Variable x: amount of grain we buy and sell
- Variable y: amount of salt we buy and sell

- Grain costs B and salt costs C; we can invest A
  - $B x + Cy \leq A$
  - Or: $Bx + Cy = A$
- We can store at most D units
  - $x + y \leq D$
- We can buy/sell at most G units of grain and H units of salt:
  - $x \leq G$
  - $y \leq H$

Formulate optimization criterion

- Maximize profit
  - Grain is bought for B and sells for E so gives $E - B$ profit
  - Salt is bought for C and sells for F so gives $F - C$ profit
- Max $(E - B)\,x \; + \; (F - C)\,y$

Linear program

- Maximize $(E - B)\,x + (F - C)\,y$
  - Subject to:
    - $B\,x + Cy \leq A$
    - $x + y \leq D$
    - $x \leq G$
    - $y \leq H$

    And
    - $x \geq 0$
    - $y \geq 0$
- Where A, B, C, D, E, F, G, H are given real numbers (or integers)
- x, y must be (non-negative) real numbers

The problem we want to solve is called a Linear Program

Universiteit Utrecht

[Faculty of Science
Information and Computing Sciences]

LP

- Optimization function:
  - Min $c_1x_1 + \ldots + c_nx_n$
  - Or Max: (this is the same, multiply c_i's with -1)
- System of lineaire constraints: we have a collection:
  - $a_{i1}x_1 + \ldots + a_{in}x_n \leq b_i$
- Usually: requirement that all variables x_i are positive:
  - $x_i \geq 0$

Transformation – equivalent formulations (= , ≤, ≥)

- Or = instead of ≤ : can be transformed easily in both directions:
  - $a_{i1}x_1 + \ldots + a_{in}x_n = b_i$ is equivalent to
    $a_{i1}x_1 + \ldots + a_{in}x_n \leq b_i$ and
    $-ai_1x_1 + \ldots + -a_{in}x_n \leq -b_i$
  - In the other direction: introduce new slack variable:
  - $a_{i1}x_1 + \ldots + a_{in}x_n + y = b_i$ and $y \geq 0$
- ≤ can be transformed to ≥ (and back) by multiplying with -1
  - $a_{i1}x_1 + \ldots + a_{in}x_n \leq b_i$
  - $-a_{i1}x_1 - \ldots - a_{in}x_n \geq -b_i$

Matrix formulation of LP

- Given are: n by m matrix A, vector c of length n, vector b of length m
- Question:
  - Find a vector x of length n, such that
  - $Ax = b$ *(multiplication of matrix and vector)*
  - x ≥ *0* *(each element in x is nonnegative)*
  - cx is as large as possible *(inner product)*

Universiteit Utrecht

[Faculty of **Science**
Information and Computing Sciences]

Use of LPs

- Many problems can be modeled as an LP
- Practical algorithms to solve LPs:
  - Simplex method (worst case exponential time, usually fast)
    - Explained later in these lectures
  - Usually slower polynomial time algorithm (ellipsoid method)
  - Commercial and freeware available programs

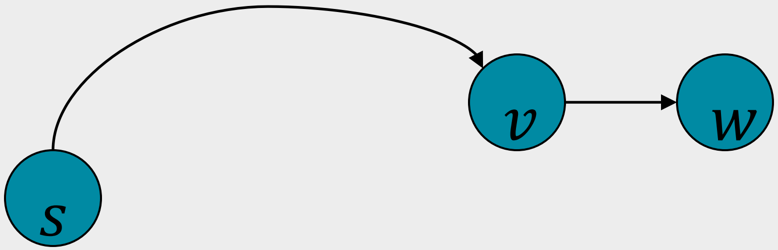
Shortest paths as LP (1)

- Given: directed graph G, each arc (v, w) has a length (positive integer) $L(v, w)$, vertex s
- Question: for each v, what is the shortest length path from s to v

Shortest paths as LP (2)

- Take for each vertex v a variable x_v.
- x_v denotes the length of the shortest path from s to v
- Take equation $x_s = 0$
- For each arc $(v, w) \in E$, take an inequality $x_v + L(v, w) \geq x_w$

- Necessary:



Shortest path as LP (3)

- But also sufficient – in the sense that minimum x-values fulfilling give the shortest paths length
- If we compute $\min \sum_{v \in V} x_v$
- then we obtain the shortest paths lengths

- Similar for some other problems (like flow)

Example: 2-commodity flow

- Suppose we have a road network (directed graph G=(V,E))
- Vertex 1 has a units of a good, say coal, which must be transported to vertex 2
- Vertex 3 has b units of a good, say oil, which must be transported to vertex 4
- All arcs in the network have a capacity: the total of the oil and the coal moved over arc (v, w) is at most $c(v, w)$.
- What arrives at a node must be sent away from the node again (except coal for 1 and 2, and oil for 3 and 4)
- Formulate as LP: can we transport all coal and oil over the network
  - Note: we do not need a cost function!

Integer Linear Programming

- Often, we need variables to be integers:
  - We can buy / build / sell / move / … only integer numbers of objects
  - Some decisions are binary: yes (1) or no (0)
- Integer Linear Program
  - All variables x_i must be non-negative integers
- Mixed ILP
  - Some variables must be integers, others real numbers

Many examples

- Logistics
- Energy
- Design
- Etc etc etc

Small and large models

- Polynomial size models … but also
- Models that are "very large"
  - Implicit descriptions of ILP
  - E.g., variable for each path in a graph, plan for one employee, etc.
  - Advanced techniques that always work with a part of the entire model (e.g., column generation)

People with reduced mobility (PRM) require assistance when travelling through the airport

http://www.schiphol.nl/Travellers/AtSchiphol/InformationForPassengersWithReducedMobility.htm

PRMs – example application

1. Management wants to find the best locations of the lounges
2. Management wants to determine the assignment of PRM's to employees such the waiting time for the PRM's outside the lounges is minimal.

- In project (by van Diepen, Utrecht), ILP techniques were used to solve ("Combinatorial optimization" techniques)
  - Other techniques: simulation, heuristics, …

Combinatorial optimization

- Define variables x representing a schedule
- Express waiting time $w(x)$ in terms of these variables
- Formulate constraints in terms of these variables: $x \in C$
- C contains a finite but very large number of elements
- Optimize:

$$\min w(x) \text{ subject to } x \in C$$

Integer linear programming

- Mainstream optimization method in scientific research
- Extremely important optimization algorithm in practice
- Used by Tennet to analyse the system adequacy (leveringszekerheid) of the Dutch electricity network
- Used by U-OV to determine the sequence of trips for each of their buses
- Record for solving Travelling Salesman Problem

LP / ILP

LP

- Variables can have real / fractional values
- Polynomial time solvable / relatively fast algorithms

ILP

- Variables must have integer values
- NP-hard / sometimes slow to solve

Another example: Testing and selling

- Three types of computers: Alpha, Beta, and Gamma.
- Net profit: \$350,- per Alpha, \$470,- per Beta, and \$610,- per Gamma.
- Every computer can be sold at the given profit.
- Testing: Alpha and Beta computers on the A-line, Gamma computers on the C-line.
- Testing takes 1 hour per computer.
- Capacity A-line: 120 hours; capacity C-line: 80 hours.
- Required labor: 10 hours per Alpha, 15 hours per Beta, and 20 hours per Gamma.
- Total amount of labor available: 2000 hours.

Example: Testing and selling

Decision variables: MA number of alpha's produced, etc

$$\max Z = 350\ \mathrm{MA} + 470\ \mathrm{MB} + 610\ \mathrm{MC}$$

Objective function

$$\begin{array}{lll} \text{subject to} & \mathrm{MA} + \mathrm{MB} & \leq 120 \quad (\text{A} - \text{line}) \\ & \mathrm{MC} & \leq 48 \quad (\text{C} - \text{line}) \\ & 10\mathrm{MA} + 15\mathrm{MB} + 20\mathrm{MC} & \leq 2000 \quad (\text{labor}) \\ & \mathrm{MA}, \mathrm{MB}, \mathrm{MC} \geq 0 & \end{array}$$

Constraints

Universiteit Utrecht

[Faculty of Science
Information and Computing Sciences]

Linear programming

Min $c^T x$

s.t. $Ax \leq b$

$x \geq 0$

With

$x \in \mathbb{R}^n$, $c \in \mathbb{Q}^n$, $\mathrm{A} \in \mathbb{Q}^{m \times n}$, and $b \in \mathbb{Q}^m$

But ….

Solution method for linear programming

- Simplex method
  - Slower than polynomial
  - Practical
- Ellipsoid method
  - Polynomial (Khachian, 1979)
  - Not practical
- Interior points methods
  - Polynomial (Karmakar, 1984)
  - Outperforms Simplex for very large instances

Seems not too hard to implement. But, for larger problems you run into numerical problems. Use a standard solver (Gurobi, CPLEX, GLPK)

$$LP \in P$$

An introduction to

INTEGER LINEAR PROGRAMS

Knapsack problem

Knapsack with volume 15
What should you take with you to maximize utility?

Item	1:paper	2:book	3:bread	4:smart -phone	5:water
Utility	8	12	7	15	12
Volume	4	8	5	2	6

Knapsack problem (2)

x_1 = 1 if item 1 is selected, 0 otherwise, x_2, ……

max $z = 8 x_1 + 12 x_2 + 7 x_3 + 15 x_4 + 12 x_5$

subject to

$$4 x_1 + 8 x_2 + 5 x_3 + 2 x_4 + 6 x_5 \leq 15$$

$$x_1, x_2, x_3, x_4, x_5 \in \{0,1\}$$

Knapsack problem (3)

- n items, knapsack volume b
- Item j has
  - Utility (revenue) c_j
  - Volume (weight) a_j
- MIP formulation
  - Decision variables: $x_j = 1$ if item j is selected and $x_j = 0$ otherwise

$$\max \sum_{j=1}^{n} c_j x_j$$

s.t.

$$\sum_{j=1}^{n} a_j x_j \le b$$

$$x_j \in \{0, 1\} \qquad (j = 1, \dots, n)$$

Binary variables

- x_i binary (in {0,1}) can be expressed as:
- $x_i \geq 0$ and $x_i \leq 1$, x_i integer

Mixed Integer linear Programming (MIP)

Min $c^T x + d^T y$

$s.t.\ Ax + By \leq b$

$x, y \geq 0$

x integral (or binary)

Extension of LP:

Success factor!

- Good news: more possibilities for modelling
  - Many problems from transportation, logistics, rostering, health care planning etc
- Bad news: larger solution times

MODELLING AS ILP'S

Modeling

- Decision variables
- Objective function
- Constraints

Assignment problem

- n persons, n jobs.
- Each person can do at most one job
- Each job has to be executed
- C_{ij} cost if person i performs job j
- We want to minimize cost

Assignment problem

- n persons, n jobs.
- Each person can do at most one job
- Each job has to be executed
- C_{ij} cost if person i performs job j
- We want to minimize cost

Polynomial time solvable with matching / flow

Modelling the assignment problem

- As ILP:
- Take binary variable x_{ij} which is 1 when person i does job j, and 0 otherwise

- At most one job per person: for each i: $\sum_j x_{ij} \leq 1$
- Each job is executed: for each j: $\sum_i x_{ij} = 1$

- Optimization: minimum cost for assignment

$$\min \sum_{ij} c_{ij} x_{ij}$$

Maximum Independent Set

- Given a graph $G = (V, E)$
- V: vertices
- E: edges
- An independent set I is a set of vertices, such that every edge has at most one vertex in I, i.e., no pair of nodes in I is adjacent.
- What is the maximum number of vertices in an independent set?

Maximum Independent Set

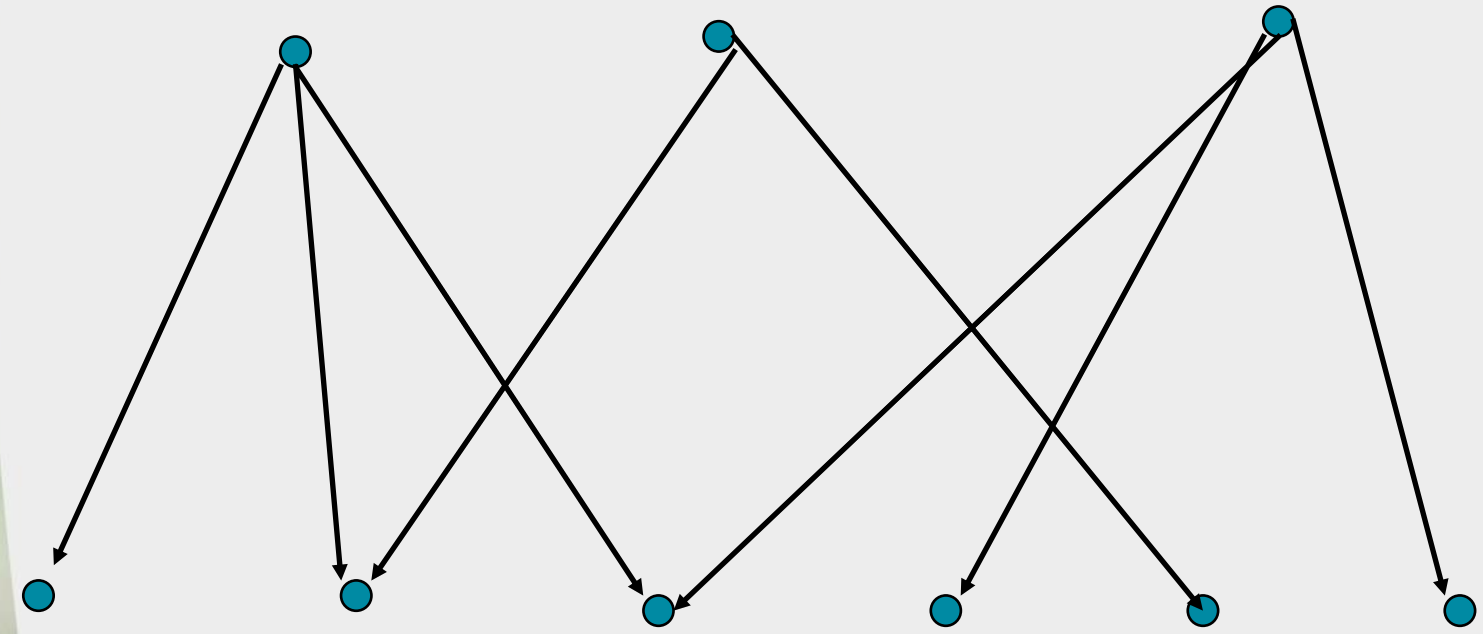
- NP-hard
- Model as ILP:

- Binary variable for each vertex v: $x_v \in \{0,1\}$:
  1 when v in the independent set, 0 otherwise
- Constraint for each edge:
  $x_v + x_w \leq 1$ for each edge $\{v,w\} \in E$
  (Check the four cases!)
- Largest independent set:

$$\max \sum_{v \in V} x_v$$

Facility location

Possible locations: n



Customers: m

Applications of facility location

- Shops, firestations, police stations, …
- Fire extinguishers in building, …
- Repairman, salesman, …
- …

Capacitated facility location problem

- Given:
  - m customers,
  - Customer demand: D_i
  - n possible locations of depots (facilities)
  - c_{ij} cost per unit product transported from depot j to customer i
  - Capacity depot: C_j
  - Fixed cost for opening depot DC: F_j
- Which depots are opened and which customer is served by which depot?

ILP model without costs for opening depots

- Non-negative integer variable x_{ij} tells how much location j serves to client i
- Constrain for each client: its demand is met
- Constraint for each location: not serving over its capacity
- Cost function: minimum cost for everything

ILP model without costs for opening depots

- Non-negative integer variable x_{ij} tells how much location j serves to client i
- Constrain for each client *i*: its demand is met

$$\sum_j x_{ij} = D_i$$

- Constraint for each location j: not serving over its capacity

$$\sum_i x_{ij} \leq C_j$$

- Cost function: minimum cost for everything

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

But now with costs for opening depots

- Binary variable y_j : 0 if not openend, 1 if opened
- Cost function is not so hard to make:
$\left(\sum_{ij} c_{ij} * x_{ij}\right) + (\sum_j F_j * y_i)$

- But … we need a constraint that tells: when a depots is not openend, then it does not serve anybody

- …

But now with costs for opening depots

- Binary variable y_j : 0 if not openend, 1 if opened
- Cost function is not so hard to make:
  $\left(\sum_{ij} c_{ij} * x_{ij}\right) + \left(\sum_j F_j * y_i\right)$

- A constraint that tells: when a depots is not openend, then it does not serve anybody
-

$$\sum_i x_{ij} + \left(1 - y_j\right) * C_i \leq C_i$$

ILP model with costs for opening depots

- Non-negative integer variable x_{ij} tells how much client i receives from location j;
  Binary variable y_j : 0 if not openend, 1 if opened
- Constraint for each client i: its demand is met: $\sum_j x_{ij} = D_i$
- Constraint for each location j: not serving over its capacity and not serving at all when closed:

$$\sum_i x_{ij} + \left(1 - y_j\right) * C_i \leq C_i$$

- Cost: $\left(\sum_{ij} c_{ij} * x_{ij}\right) + \left(\sum_j F_j * y_i\right)$

Universiteit Utrecht

[Faculty of Science
Information and Computing Sciences]

Capacitated facility location:

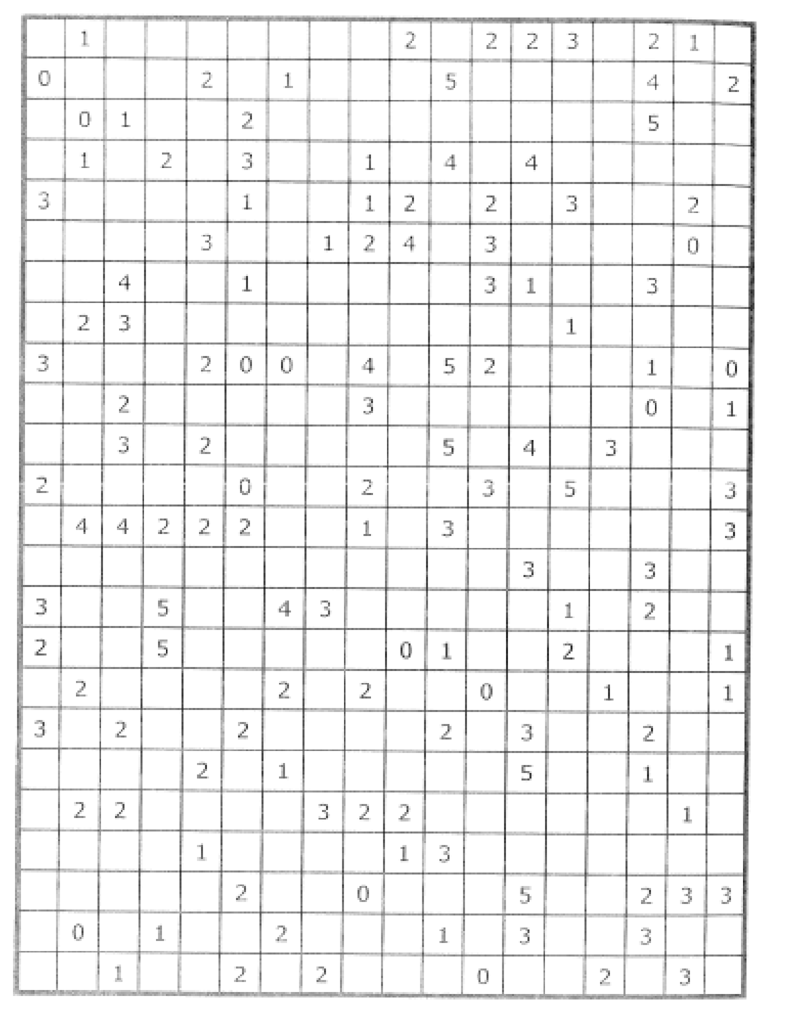
- Our example shows modelling possibilities with binary variables
- Our model uses binary variables for ***fixed cost***
- Our model uses binary variables ***forcing constraints***:
  - depot can only be used when it is open.

- Variations with different models …

Treasure island

- Diamonds are buried on an island
- Numbers give number of diamonds in neighboring positions (include diagonal)
- At most one diamond per position
- No diamond at position with number

- We now have a feasibility problem



Feasibility LP

- No objective function, or just: max 0
- Boolean ariables: $x_{(i,j)}$ which is 1 when there is a diamond at position (i,j)
- For each number, there is a constraint usually of the form
- $x_{(i-1,j-1)} + x_{(i-1,j)} + x_{(i-1,j+1)} + x_{(i,j-1)} + x_{(i,j+1)} + x_{(i-1,j-1)} + x_{(i-1,j)} + x_{(i-1,j+1)} = c$
  with c the number at position (i,j)

Challenge: Treasure island with pitfall

- Like treasure island but exactly one given number is incorrect.

How to solve the challenge – Plan 1 fails

- Add for each number on (i, j) a variable $w_{i,j}$ which is 1 when the value at (i, j) is wrong and 0 when the value is correct
- A constraint that tells that exactly one given number is wrong (summarizing over all pairs (i, j) with a number)

$$\sum_{(i,j)} w_{i,j} = 1$$

- Check for numbers allowing to be wrong – not easy to model…

How to solve the challenge – Plan 2

- Add for each number on (i,j) a variable $s_{(i,j)}$ which is 1 when the value at (i,j) is too small and 0 when the value is correct
- Add for each number on (i,j) a variable $g_{(i,j)}$ which is 1 when the value at (i,j) is too large and 0 when the value is correct
- A constraint that tells that exactly one given number is wrong (summarizing over all (i,j) with a number)

$$\sum_{(i,j)} (s_{(i,j)} + g_{(i,j)}) = 1$$

- Check for numbers allowing to be wrong – we can do that now…

Check that numbers are correct

- Suppose on position (i,j) we have the number $c \in \{0,1,2,3,4,5,6,7,8\}$
- $x_{(i-1,j-1)} + x_{(i-1,j)} + x_{(i-1,j+1)} + x_{(i,j-1)} + x_{(i,j+1)} + x_{(i-1,j-1)} + x_{(i-1,j)} + x_{(i-1,j+1)} = c$
- becomes
- $x_{(i-1,j-1)} + x_{(i-1,j)} + x_{(i-1,j+1)} + x_{(i,j-1)} + x_{(i,j+1)} + x_{(i-1,j-1)} + x_{(i-1,j)} + x_{(i-1,j+1)} - 8 * g_{(i,j)} \leq c$ together with
- $x_{(i-1,j-1)} + x_{(i-1,j)} + x_{(i-1,j+1)} + x_{(i,j-1)} + x_{(i,j+1)} + x_{(i-1,j-1)} + x_{(i-1,j)} + x_{(i-1,j+1)} + 8 * s_{(i,j)} \geq c$

SOLVING ILPS

Relaxation

(Mixed) Integer linear program

Min $c^T x$

$s.t.\ Ax + By \leq b$

$x, y \geq 0$

x integral

(or binary)

LP-relaxation

Min $c^T x$

$s.t.\ Ax + By \leq b$

$x, y \geq 0$

Lower bound (or upper bound in case of maximization)

Universiteit Utrecht

[Faculty of **Science**
Information and Computing Sciences]

Modelling choice matters: Strength (quality) of an ILP (or MIP) formulation

- Consider some ILP problem, with T the set of feasible integral solutions
- We can have two different ILP models, with relaxations
- For a formulation A, write P_A as the feasible set of solutions of the relaxation
- Ideal situation: P_F is the convex hull of T
- Formulation A is stronger than formulation B if

$$P_A \subset P_B$$

- Hence, the bound from model A is stronger
- Most likely, solving the MIP by branch-and-bound will be faster

Modelling choice matters !!!

Universiteit Utrecht

[Faculty of **Science**
Information and Computing Sciences]

Modelling choice matters:

If you have a MIP formulation and it solves very slow

- This may not be the final answer
- You might try an alternative formulation

- Try different things and test them!
- Be creative!

Relaxations

Mixed Integer Linear Program

Min $c^T x$

$s.t.\ Ax + By \leq b$

$x, y \geq 0$

x integral

(or binary)

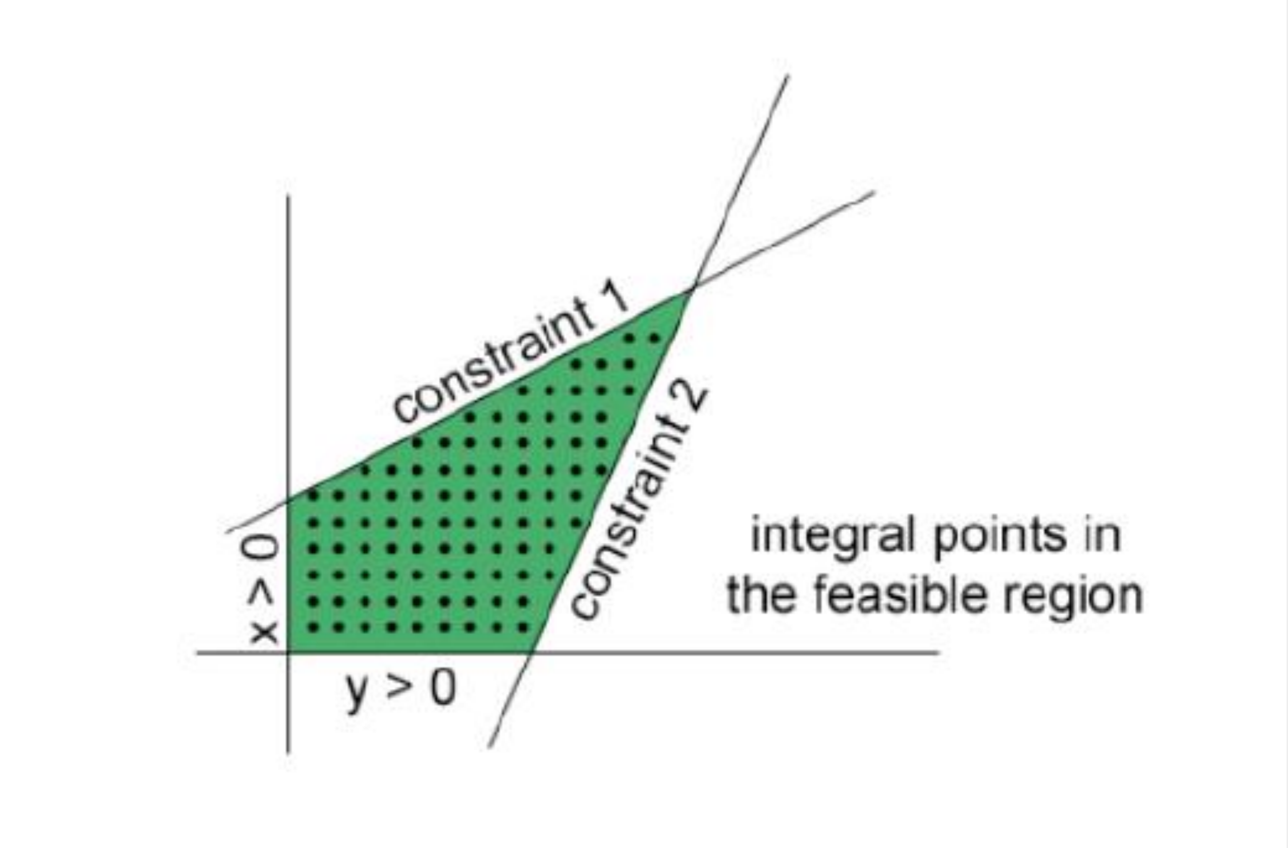
LP-relaxation

Min $c^T x$

$s.t.\ Ax + By \leq b$

$x, y \geq 0$

Lower bound (or upper bound in case of maximization)



constraint 1
constraint 2
x > 0
y > 0
integral points in
the feasible region

Knapsack problem

- n items, knapsack volume b
- Item j has
  - Utility (revenue) c_j
  - Volume (weight) a_j
- ILP formulation
  - Decision variables: $x_j = 1$ if item j is selected and $x_j = 0$ otherwise

$$\max \sum_{j=1}^{n} c_j x_j$$

s.t.

$$\sum_{j=1}^{n} a_j x_j \leq b$$

$$x_j \in \{0, 1\} \qquad (j = 1, \dots, n)$$

Knapsack problem: LP-relaxation

- LP-relaxation: Greedy algorithm

Step 0. Order variables such that $\frac{c_1}{a_1} \geq \frac{c_2}{a_2} \geq \ldots \geq \frac{c_n}{a_n}$

Step 1. $x_i \leftarrow 0\ \forall_i;$ restcapacity $\bar{b} = b; i = 1$

Step 2. If $a_i \leq \bar{b}$, then $x_i \leftarrow 1,$ else $x_i \leftarrow \frac{\bar{b}}{a_i}$. Set $\bar{b} \leftarrow \bar{b} - a_i x_i; i \leftarrow i + 1$

Step 3. If $\bar{b} > 0,$ go to Step 2.

The greedy algorithm gives the optimum for the relaxation (Algoritmiek)

- Feasible solution: rounding down

How to solve an Integer Linear Program

- First solve the LP-relaxation
  - Simplex method was discussed
  - There is excellent software for this (even Excel can solve a (not too large) LP)
- If LP-relaxation has integral solution: finished ☺ ☺
- Otherwise, proceed by branch-and-bound

Solving MIP by branch-and-bound: informally (for maximization)

Split into sub problems, e.g. by fixing a variable to 0 or 1. This is *branching*, you get nodes of a tree.

Select a node to evaluate. You can compute:

- upper bound by solving LP-relaxation
- a feasible solution (e.g. by rounding heuristic)

4 options:

1. Infeasible: do not search further from this node
2. LP-relaxation upper bound less than or equal to the best known feasible solution: hopeless node, do not search further here *(bounding)*
3. LP-relaxation has integral solution: this node is completely solved. ☺
4. LP-relaxation upper bound larger than heuristic solution. Search further by splitting (*branching)*

Solving MIP by branch-and-bound

Let x* be the best-known feasible solution and let v(x*) be its objective value.

1. Select an active sub problem F_i (unevaluated node)
2. Solve LP-relaxation of F_i. If F_i is infeasible: delete node and go to 1
3. Consider upper bound $Z_{LP}(F_i)$ from LP-relaxation and compute feasible solution x_f (e.g. by rounding)
   1. If $Z_{LP}(F_i) \leq v(x)^*$ delete node
   2. If $v(x_f) > v(x^*)$: update x*
   3. If solution x_{LP} to LP-relaxation is integral, then node finished, otherwise split node into two new active sub problems
4. Go to step 1

Optional

This is for maximization problem.

Universiteit Utrecht

[Faculty of Science
Information and Computing Sciences]

Branch and bound

- Different setups:
  - Working with “pool of subproblems”
    - Generate some “best solution so far” (e.g., run heuristic)
    - Repeat till empty pool
      - Take subproblem from pool
      - Handle subproblem: generate upper bound, lower bound
        - If better solution than best solution so far: update best solution so far
        - Bound can tell: no better solution in this branch: discard subproblem
        - LP gives integer solution: optimum of this branch found
        - Otherwise: take variable, and branch – generate (at least 2) new subproblems
  - Recursive algorithm
    - Similar, but subproblems are generated by recursive call
  - Methods can be combined with memorization (Dynamic Programming)

An example
Branch and bound for Knapsack

- Knapsack with capacity 15
- Items with volume and utility

	1	2	3	4	5
Volume a_i	4	8	3	6	5
Utility c_i	8	12	7	15	12
c_i/a_i	2	1 1/2	2 1/3	2 1/2	2 2/5

Branch and bound for Knapsack

- Knapsack with capacity 15
- Items with volume and utility

	1	2	3	4	5
Volume a_i	4	8	3	6	5
Utility c_i	8	12	7	15	12
c_i/a_i	2	1 1/2	2 1/3	2 1/2	2 2/5

- $\max 8x_1 + 12x_2 + 7x_3 + 15x_4 + 12x_5$
  - Such that $4x_1 + 8x_2 + 3x_3 + 6x_4 + 5x_5 \leq 15$
  - And $x_1 \in \{0,1\}$, $x_2 \in \{0,1\}$, $x_3 \in \{0,1\}$, $x_4 \in \{0,1\}$, $x_5 \in \{0,1\}$

Start

- Subproblem = main problem
- Solve relaxation:
- $\max 8x_1 + 12x_2 + 7x_3 + 15x_4 + 12x_5$
  - Such that $4x_1 + 8x_2 + 3x_3 + 6x_4 + 5x_5 \leq 15$
  - And $x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0,\ x_4 \geq 0,\ x_5 \geq 0,\ x_1 \leq 1, x_2 \leq 1, x_3 \leq 1, x_4 \leq 1, x_5 \leq 1$
- Theory tells that greedy (wrt c_i/a_i) gives optimal solution of relaxation:
- $x_1 = \frac{1}{4},\ x_2 = 0,\ x_3 = 1,\ x_4 = 1,\ x_5 = 1,$

	1	2	3	4	5
Volume a_i	4	8	3	6	5
Utility c_i	8	12	7	15	12
c_i/a_i	2	1 1/2	2 1/3	2 1/2	2 2/5

Start

- $\max 8x_1 + 12x_2 + 7x_3 + 15x_4 + 12x_5$
  - Such that $4x_1 + 8x_2 + 3x_3 + 6x_4 + 5x_5 \leq 15$
  - And $x_1 \geq 0$, $x_2 \geq 0$, $x_3 \geq 0$, $x_4 \geq 0$, $x_5 \geq 0$, $x_1 \leq 1, x_2 \leq 1, x_3 \leq 1, x_4 \leq 1, x_5 \leq 1$
- Optimal solution of relaxation:
  $x_1 = \frac{1}{4}$, $x_2 = 0$, $x_3 = 1$, $x_4 = 1$, $x_5 = 1$
- Upper bound (value of relaxation): 36
- Lower bound: 34= 7+15+12
  (rounded solution items 3, 4, 5)

	1	2	3	4	5
Volume a_i	4	8	3	6	5
Utility c_i	8	12	7	15	12
c_i/a_i	2	1 1/2	2 1/3	2 1/2	2 2/5

Branch

- Optimal solution of relaxation main problem
  $x_1 = \frac{1}{4}, x_2 = 0, x_3 = 1, x_4 = 1, x_5 = 1$
- Upper bound: 36
- Lower bound: 34= 7+15+12
  (rounded solution items 3, 4, 5)
- We now can branch on some item. In the example, we decide to branch on item 5

- Subproblem 1: 5 is not taken: $x_5 = 0$
- Subproblem 2: 5 is taken: $x_5 = 1$

	1	2	3	4	5
Volume a_i	4	8	3	6	5
Utility c_i	8	12	7	15	12
c_i/a_i	2	1 1/2	2 1/3	2 1/2	2 2/5

Subproblem 1

- $\max 8x_1 + 12x_2 + 7x_3 + 15x_4 + 12x_5$
  - Such that $4x_1 + 8x_2 + 3x_3 + 6x_4 + 5x_5 \leq 15$
  - And $x_1 \in \{0,1\}$, $x_2 \in \{0,1\}$, $x_3 \in \{0,1\}$, $x_4 \in \{0,1\}$, $x_5 = 0$
- Relaxation:
- $\max 8x_1 + 12x_2 + 7x_3 + 15x_4 + 12x_5$
  - Such that $4x_1 + 8x_2 + 3x_3 + 6x_4 + 5x_5 \leq 15$
  - And $x_1 \geq 0$, $x_2 \geq 0$, $x_3 \geq 0$, $x_4 \geq 0$, $x_5 = 0, x_1 \leq 1, x_2 \leq 1, x_3 \leq 1, x_4 \leq 1$. Or, equivalently:
- $\max 8x_1 + 12x_2 + 7x_3 + 15x_4$
  - Such that $4x_1 + 8x_2 + 3x_3 + 6x_4 \leq 15$
  - And $x_1 \geq 0$, $x_2 \geq 0$, $x_3 \geq 0$, $x_4 \geq 0, x_1 \leq 1, x_2 \leq 1, x_3 \leq 1, x_4 \leq 1$

Subproblem 1

- $\max 8x_1 + 12x_2 + 7x_3 + 15x_4$
  - Such that $4x_1 + 8x_2 + 3x_3 + 6x_4 \leq 15$
  - And $x_1 \geq 0$, $x_2 \geq 0$, $x_3 \geq 0$, $x_4 \geq 0$, $x_1 \leq 1, x_2 \leq 1, x_3 \leq 1, x_4 \leq 1$
- Has solution: $x_1 = 1, x_2 = \frac{1}{4}, x_3 = 1, x_4 = 1$
- Value of relaxation: 33
- We already had a `best solution' with value 34 (items 3, 4, 5), so we see that setting $x_5 = 0$ cannot improve the solution: this branch is finished

Subproblem 2

- $\max 8x_1 + 12x_2 + 7x_3 + 15x_4 + 12x_5$
  - Such that $4x_1 + 8x_2 + 3x_3 + 6x_4 + 5x_5 \leq 15$
  - And $x_1 \in \{0,1\}$, $x_2 \in \{0,1\}$, $x_3 \in \{0,1\}$, $x_4 \in \{0,1\}$, $x_5 = 1$
- Relaxation:
- $\max 8x_1 + 12x_2 + 7x_3 + 15x_4 + 12x_5$
  - Such that $4x_1 + 8x_2 + 3x_3 + 6x_4 + 5x_5 \leq 15$
  - And $x_1 \geq 0$, $x_2 \geq 0$, $x_3 \geq 0$, $x_4 \geq 0$, $x_5 = 1$, $x_1 \leq 1, x_2 \leq 1, x_3 \leq 1, x_4 \leq 1$
- Optimal solution of relaxation is $x_1 = \frac{1}{4}$, $x_2 = 0$, $x_3 = 1$, $x_4 = 1$, $x_5 = 1$
- Value 36, rounded solution 34, so we need to branch again on this subproblem
- Say, we branch on setting $x_4 = 0$ or $x_4 = 1$ (subproblems 3 and 4)

Subproblem 3: $x_4 = 0, x_5 = 1$

- $\max 8x_1 + 12x_2 + 7x_3 + 15x_4 + 12x_5$
  - Such that $4x_1 + 8x_2 + 3x_3 + 6x_4 + 5x_5 \leq 15$
  - And $x_1 \in \{0,1\}$, $x_2 \in \{0,1\}$, $x_3 \in \{0,1\}$, $x_4 = 0$, $x_5 = 1$
- Relaxation:
- $\max 8x_1 + 12x_2 + 7x_3 + 15x_4 + 12x_5$
  - Such that $4x_1 + 8x_2 + 3x_3 + 6x_4 + 5x_5 \leq 15$
  - And $x_1 \geq 0$, $x_2 \geq 0$, $x_3 \geq 0$, $x_4 = 0$, $x_5 = 1$, $x_1 \leq 1, x_2 \leq 1, x_3 \leq 1$
- Solution of relaxation:
  $x_1 = 1$, $x_2 = \frac{3}{8}$, $x_3 = 1$, $x_4 = 0$, $x_5 = 1$
- Value of relaxation: $31\frac{1}{2}$: worse than best known integral solution: stop this branch

Subproblem 4: $x_4 = 1, x_5 = 1$

- $\max 8x_1 + 12x_2 + 7x_3 + 15x_4 + 12x_5$
  - Such that $4x_1 + 8x_2 + 3x_3 + 6x_4 + 5x_5 \leq 15$
  - And $x_1 \in \{0,1\}$, $x_2 \in \{0,1\}$, $x_3 \in \{0,1\}$, $x_4 = 1$, $x_5 = 1$
- Relaxation:
- $\max 8x_1 + 12x_2 + 7x_3 + 15x_4 + 12x_5$
  - Such that $4x_1 + 8x_2 + 3x_3 + 6x_4 + 5x_5 \leq 15$
  - And $x_1 \geq 0$, $x_2 \geq 0$, $x_3 \geq 0$, $x_4 = 1$, $x_5 = 1$, $x_1 \leq 1, x_2 \leq 1, x_3 \leq 1$
- Solution of relaxation:
  $x_1 = \frac{1}{4}$, $x_2 = 0$, $x_3 = 1$, $x_4 = 1$, $x_5 = 1$
- Again: we need to branch (relaxation has value 36, and rounded solution still gives items 3, 4, 5 with value 34)
- We branch now on item 3: $x_3 = 0$ or $x_3 = 1$ (subproblems 5 and 6)

Universiteit Utrecht

[Faculty of **Science**
Information and Computing Sciences]

Subproblem 5: $x_3 = 0, x_4 = 1, x_5 = 1$

- $\max 8x_1 + 12x_2 + 7x_3 + 15x_4 + 12x_5$
  - Such that $4x_1 + 8x_2 + 3x_3 + 6x_4 + 5x_5 \leq 15$
  - And $x_1 \in \{0,1\}$, $x_2 \in \{0,1\}$, $x_3 = 0$, $x_4 = 1$, $x_5 = 1$
- Relaxation:
- $\max 8x_1 + 12x_2 + 7x_3 + 15x_4 + 12x_5$
  - Such that $4x_1 + 8x_2 + 3x_3 + 6x_4 + 5x_5 \leq 15$
  - And $x_1 \geq 0$, $x_2 \geq 0$, $x_3 = 0$, $x_4 = 1$, $x_5 = 1$, $x_1 \leq 1, x_2 \leq 1$
- Solution of relaxation:
  $x_1 = 1$, $x_2 = 0$, $x_3 = 0$, $x_4 = 1$, $x_5 = 1$
- Value of this relaxation is 35
- We found a better solution – keep this one now! (35>34)
- The relaxation has integral solution: optimal solution of this branch – this branch is finished!

Subproblem 6: $x_3 = 1, x_4 = 1, x_5 = 1$

- $\max 8x_1 + 12x_2 + 7x_3 + 15x_4 + 12x_5$
  - Such that $4x_1 + 8x_2 + 3x_3 + 6x_4 + 5x_5 \leq 15$
  - And $x_1 \in \{0,1\}$, $x_2 \in \{0,1\}$, $x_3 = 1$, $x_4 = 1$, $x_5 = 1$
- Relaxation:
- $\max 8x_1 + 12x_2 + 7x_3 + 15x_4 + 12x_5$
  - Such that $4x_1 + 8x_2 + 3x_3 + 6x_4 + 5x_5 \leq 15$
  - And $x_1 \geq 0$, $x_2 \geq 0$, $x_3 = 1$, $x_4 = 1$, $x_5 = 1$, $x_1 \leq 1, x_2 \leq 1$
- Solution of relaxation:
  $x_1 = \frac{1}{4}$, $x_2 = 0$, $x_3 = 1$, $x_4 = 1$, $x_5 = 1$
- Again: we need to branch (relaxation has value 36, and rounded solution still gives items 3, 4, 5 with value 34)
- We branch now on item 1: $x_1 = 0$ or $x_1 = 1$ (subproblems 7 and 8)

Universiteit Utrecht

[Faculty of Science
Information and Computing Sciences]

Subproblem 7: $x_1 = 0, x_3 = 1, x_4 = 1, x_5 = 1$

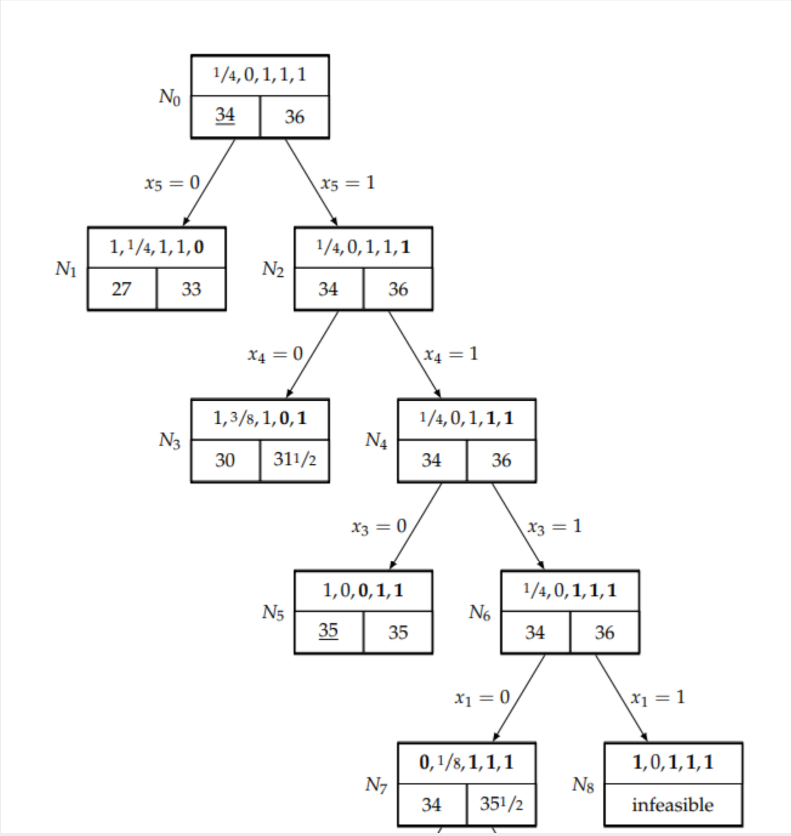
- $\max 8x_1 + 12x_2 + 7x_3 + 15x_4 + 12x_5$
  - Such that $4x_1 + 8x_2 + 3x_3 + 6x_4 + 5x_5 \leq 15$
  - And $x_1 = 0$, $x_2 \in \{0,1\}$, $x_3 = 1$, $x_4 = 1$, $x_5 = 1$
- Relaxation:
- $\max 8x_1 + 12x_2 + 7x_3 + 15x_4 + 12x_5$
  - Such that $4x_1 + 8x_2 + 3x_3 + 6x_4 + 5x_5 \leq 15$
  - And $x_1 = 0$, $x_2 \geq 0$, $x_3 = 1$, $x_4 = 1$, $x_5 = 1, x_2 \leq 1$
- Solution of relaxation:
  $x_1 = 0$, $x_2 = \frac{1}{8}$, $x_3 = 1$, $x_4 = 1$, $x_5 = 1$
- The relaxation has value $35\frac{1}{2}$: we can stop this branch, as we never get an integral solution with value better than $\left\lfloor 35\frac{1}{2} \right\rfloor = 35$ (which equals or best solution)

Subproblem 8: $x_1 = 1, x_3 = 1, x_4 = 1, x_5 = 1$

- $\max 8x_1 + 12x_2 + 7x_3 + 15x_4 + 12x_5$
  - Such that $4x_1 + 8x_2 + 3x_3 + 6x_4 + 5x_5 \leq 15$
  - And $x_1 = 1$, $x_2 \in \{0,1\}$, $x_3 = 1$, $x_4 = 1$, $x_5 = 1$
- Relaxation:
- $\max 8x_1 + 12x_2 + 7x_3 + 15x_4 + 12x_5$
  - Such that $4x_1 + 8x_2 + 3x_3 + 6x_4 + 5x_5 \leq 15$
  - And $x_1 = 1$, $x_2 \geq 0$, $x_3 = 1$, $x_4 = 1$, $x_5 = 1$, $x_2 \leq 1$
- This subproblem is infeasible (if we take items 1, 3, 4 and 5 the total weight is 4+3+6+5=18>15)
- So, we stop this branch

Wrapup example

- Now, all subproblems have been handled.
- The best solution that we found is the optimal one: items 1, 4, 5 with value 35



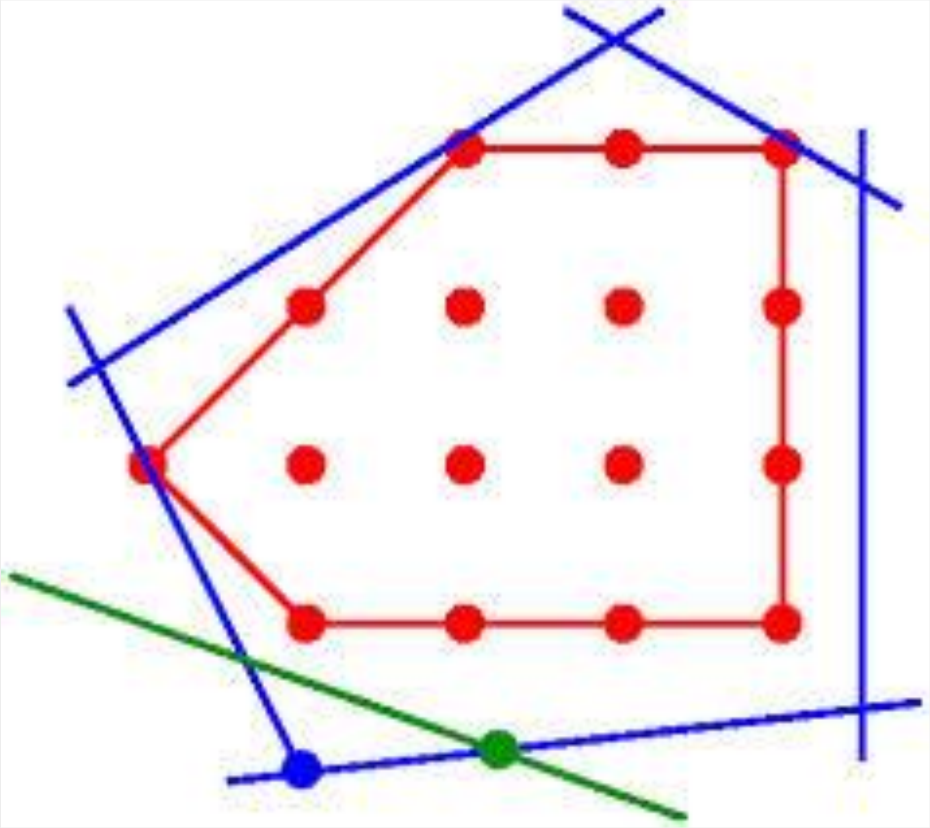
Universiteit Utrecht

[Faculty of Science
Information and Computing Sciences]

Advanced technique: Cutting plane

- Adding additional inequalities – sometimes coming from application, sometimes from looking at the integrality of solutions

Cutting plane algorithm



Cutting plane algorithm

1. Solve the LP-relaxation. Let x^* be an optimal solution of the relaxation.
2. If x^* is integral, stop: x^* is an optimal solution to the integer linear programming problem.
3. If not, add a *valid inequality* that is not satisfied by x^* and go to Step 1. To find such an inequality you have to solve a separation problem.

Valid inequality: linear constraint that is satisfied by all integral solutions.

Valid inequalities

- General
  - Gomory cuts
- Problem specific

Solving LP-relaxation

- The Simplex method gives two types of variables:
  - $x_i \in B$: basic variables, one for each constraint, can be non-zero (left-side in Dictionary)
  - $x_i \in N$: non-basic variables, are zero (right-side in Dictionary)
  - during the algorithm B changes step by step
- When running the Simplex method you find the following type of equations (are rows of the dictionary)
  - for each $x_i \in B$:

$$x_i = \bar{a}_{io} - \sum_{j \in N} \bar{a}_{ij} x_j$$

$$\Leftrightarrow$$

$$x_i + \sum_{j \in N} \bar{a}_{ij} x_j = \bar{a}_{io}$$

Dictionary: terminology from Simplex Algorithm – equations -> rows in matrix

Example

$$\max 2x_1 + x_2 \text{ subject to}$$
$$x_1 - x_2 \leq 1$$
$$2x_1 + 2x_2 \leq 7$$
$$x_1, x_2 \geq 0$$

$$\Leftrightarrow$$

$$\max 2x_1 + x_2 \text{ subject to}$$
$$x_1 - x_2 + x_3 = 1$$
$$2x_1 + 2x_2 + x_4 = 7$$
$$x_1, x_2, x_3, x_4 \geq 0$$

- Final dictionary:

$$z = 5\frac{3}{4} - \frac{1}{2}x_3 - \frac{3}{4}x_4$$
$$x_1 = 2\frac{1}{4} - \frac{1}{2}x_3 - \frac{1}{4}x_4$$
$$x_2 = \frac{5}{4} + \frac{1}{2}x_3 - \frac{1}{4}x_4$$

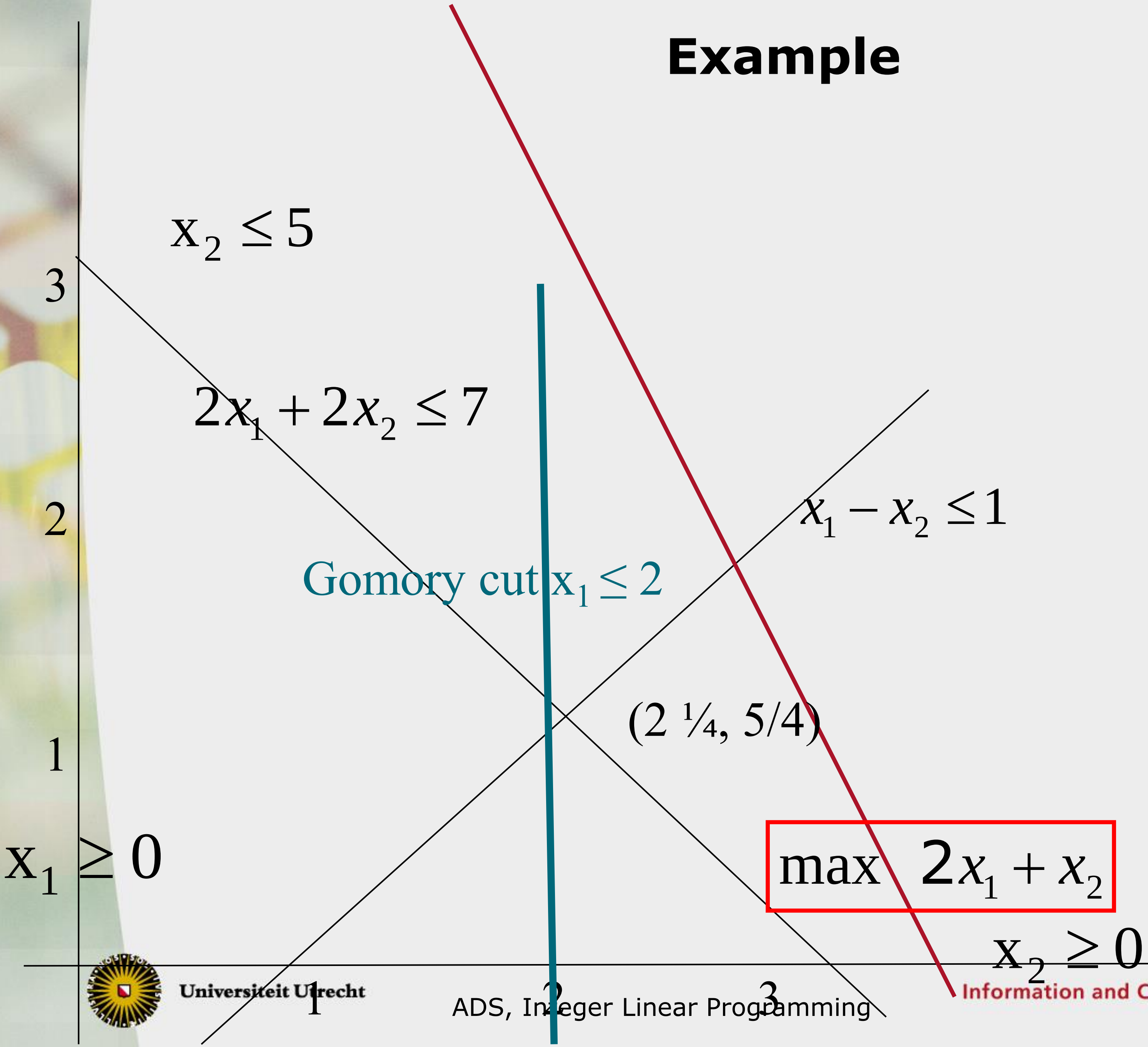
Dictionary: terminology from Simplex Algorithm – equations -> rows in matrix

Example

- $x_2 - \frac{1}{2}x_3 + \frac{1}{4}x_4 = \frac{5}{4}$
- $x_2 - x_3 \leq \frac{5}{4}$
- We look for an integral solution, so we know even that
- $x_2 - x_3 \leq 1$

- This can be combined with $x_1 - x_2 + x_3 = 1$
  - Hence: $x_2 - (1 - x_1 + x_2) \leq 1 \Leftrightarrow \mathrm{x}_1 \leq 2$

Example



$x_2 \leq 5$
$2x_1 + 2x_2 \leq 7$
$x_1 - x_2 \leq 1$
Gomory cut $x_1 \leq 2$
(2 ¼, 5/4)
$\boxed{\max \; 2x_1 + x_2}$
$x_1 \geq 0$
$x_2 \geq 0$
1
2
3

Gomory cuts

- If solution is fractional, then at least one of the $\bar{a}_{i0}$ is fractional
- Take row from final dictionary corresponding to fractional basic variable

$$x_i + \sum_{j \in N} \bar{a}_{ij} x_j = \bar{a}_{io} \text{ with } \bar{a}_{io} \text{ fractional}$$

- Gomory cut

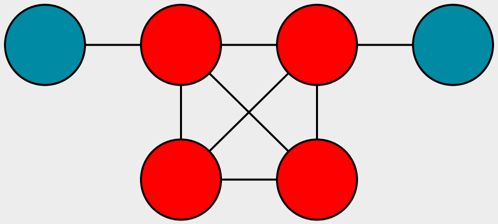
$$x_i + \sum_{j \in N} \lfloor \bar{a}_{ij} \rfloor x_j \leq \lfloor \bar{a}_{io} \rfloor$$

Problem-specific valid inequalities

- Usually classes of inequalities
- Class of inequalities: set of inequalities of a specific form

- *Finding valid inequalities is a combinatorial challenge*

Problem-specific valid inequalities: An example: Independent Set



- A clique in a graph is a set of vertices that all have an edge to each other
- At most one vertex of a clique is in the independent set
- For the independent set, we had a variable x_v for each vertex v, with $x_v = 1$ when v in the independent set
- Now, we can add for each clique W (known to us) an inequality: $\sum_{v \in W} x_v \leq 1$

[Faculty of **Science**
Information and Computing Sciences]

Universiteit Utrecht

Branch-and-cut

- Combination of cutting planes and MIP solving by branch-and-bound
- Applied in well-known MIP-solvers:
  - CPLEX
  - GUROBI
  - GNU Linear Programming Kit (**GLPK**)
  - COIN-OR
- World-record exact TSP solving
  - CONCORDE: http://www.math.uwaterloo.ca/tsp/concorde/index.html

Solving MIP by branch-and-bound or branch-and-cut

Let x* be the best known feasible solution

1. Select an active sub problem F_i (unevaluated node)
2. If F_i is infeasible: delete node
3. Compute upper bound $Z_{LP}(F_i)$ by solving LP-relaxation and adding *cutting planes*. Find feasible solution x_f (e.g. by rounding)

   If $Z_{LP}(F_i) \leq$ value x* delete node (bounding)

   If x_f is better than x*: update x*

   If solution x_{LP} to LP-relaxation is integral,

   then If x_{LP} is better than x*: update x* and node finished,

   otherwise split node into two new subproblems (branching)
4. Go to step 1

Search strategy

Primal heuristic

How many cuts?
Which classes?

Branching strategy

Branch-and-cut: choices

- Search strategy:
  - x=1 before x=0, other the other way around
  - Depth first
  - Breadth first
  - Best bound: for maximization go to the node with the largest LP-bound

- Cut generation:
  - As many cuts as possible
  - All cuts in root node, nothing in the remainder of the tree
  - All cuts in root node, only a subset from the classes in the remainder of the tree

Branch-and-cut: choices (2)

- Primal heuristic, usually based on rounding LP solution
- Branching strategy:
  - Branch on fractional variables closest to ½
  - Branch on fractional variables closest to 1
  - Branch on important variable (ratio in knapsack)
  - SOS-branching:
    - Special Ordered Sets (SOS): a set of variables, at most one of which can take a non-zero value
    - Let S be a Special Ordered Set.
    - Nodes $\sum_{j \in S} x_j = 1$ and $\sum_{j \in S} x_j = 0$

Branch-and-cut is a framework algorithm!!

Last century, tailoring to your own problem was necessary in most cases and a huge amount of research has been undertaken in doing this:

- Pre-processing
- Classes of valid inequalities
- How many cuts in each node?
- Search strategy
- Branching strategy
- Primal heuristics
- Reduced cost fixing

Well-known MIP solvers like CPLEX and Gurobi successfully (and secretly) apply a lot of the above techniques.

- **Tailoring of branch-and-cut sometimes successful, especially valid inequalities in case of a weak LP-relaxations**

MIP solvers

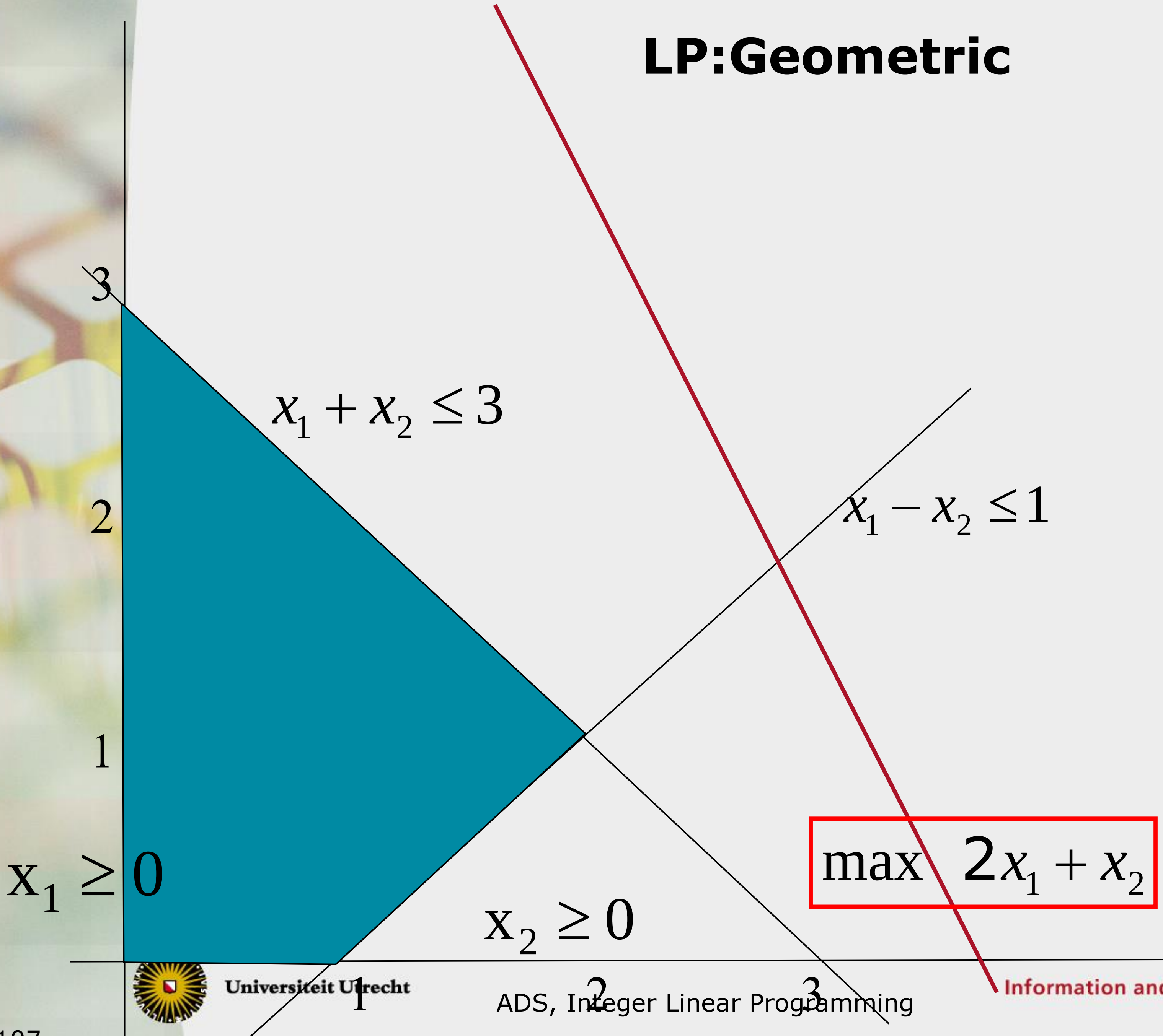
- CPLEX
- GUROBI
- GNU Linear Programming Kit (**GLPK**)
- COIN-OR

- GLPK and COIN-OR are open source,
- CPLEX and GUROBI are commercial but free for academic purposes
- GUROBI recommended

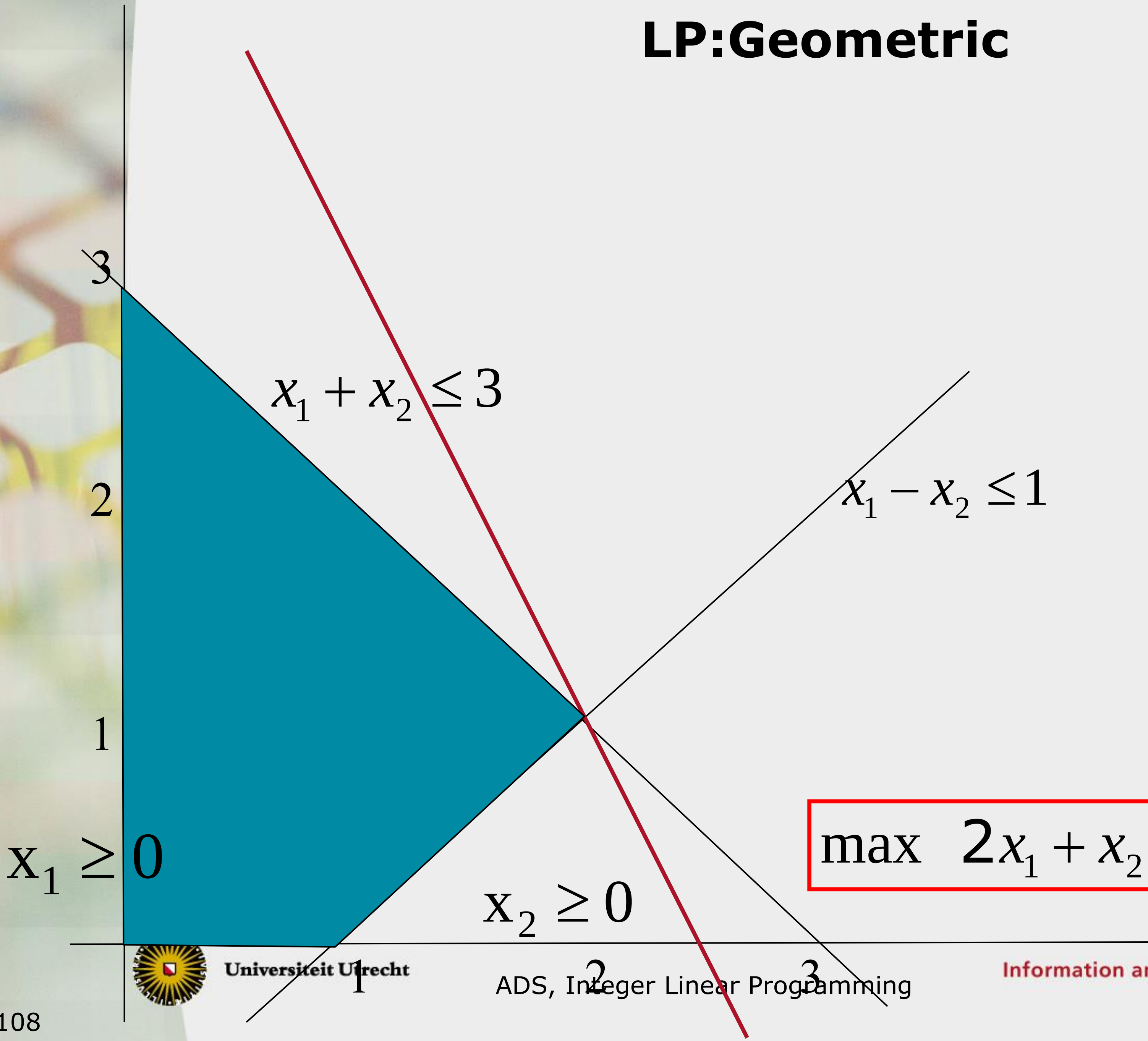
Still we are talking about combinatorial optimization

- We have many, many, many variables
- Still large computation times
- Decomposition approaches often help
- Algorithmic challenges: you need clever algorithms that every now and then use LP-solvers as a component
- **More in**
  - **Scheduling and timetabling**

LP:Geometric



LP:Geometric

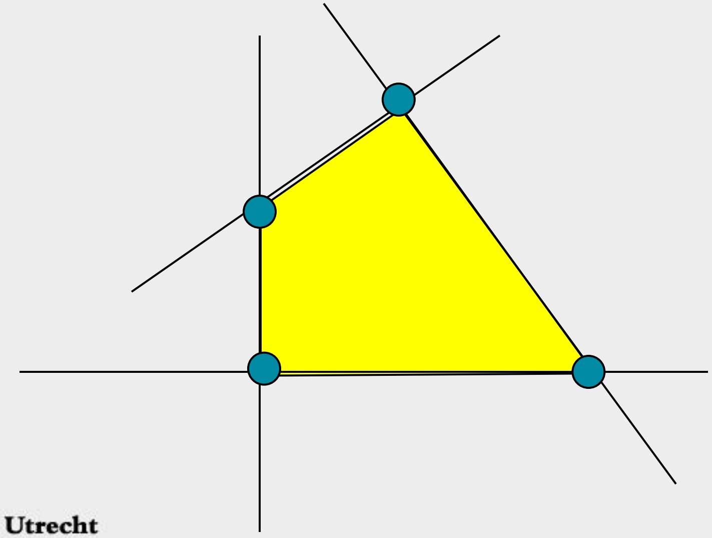


$\boxed{\max \ 2x_1 + x_2}$

The Simplex method (sketch)

- Solves an LP
- Can use exponential time in worst case, but often fast

- Optimum (if existing) is in a “vertex” (corner)
- Move from vertex to “better” vertex



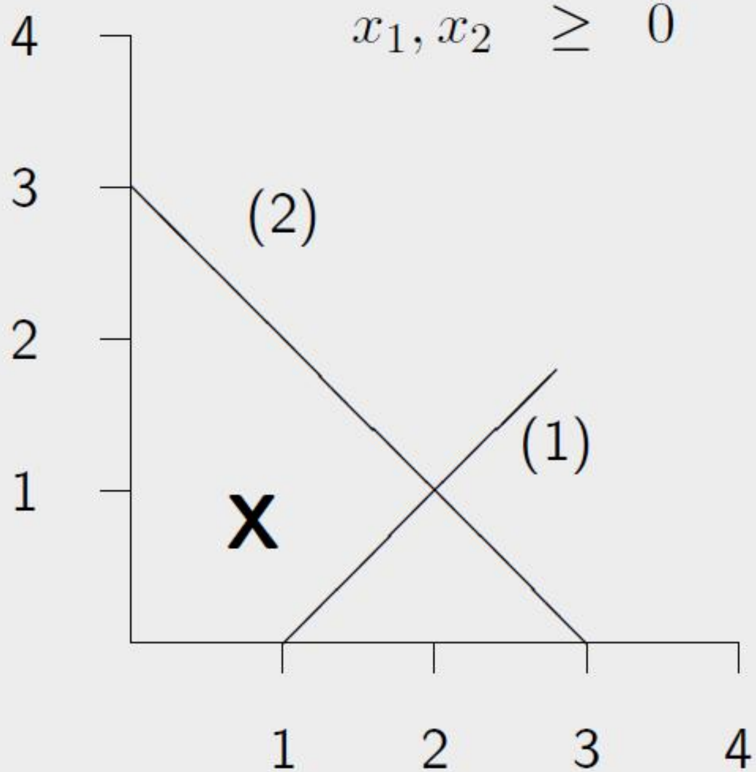
Universiteit Utrecht

[Faculty of Science
Information and Computing Sciences]

Feasible region

Constraints describing the feasible region $\mathbf{X}$:

$$\begin{aligned} x_1 - x_2 &\leq 1 \\ x_1 + x_2 &\leq 3 \\ x_1, x_2 &\geq 0 \end{aligned}$$



$$\begin{aligned} (1):\quad x_1 - x_2 &= 1 \\ (2):\quad x_1 + x_2 &= 3 \end{aligned}$$

Solving the LP

Use the objective of maximizing $z = 2x_1 + x_2$.

- Introduce slack variables $x_3, x_4 \geq 0$
- New description feasible region:
$$\begin{aligned} x_1 - x_2 + x_3 &= 1 \\ x_1 + x_2 + x_4 &= 3 \\ x_1, x_2, x_3, x_4 &\geq 0 \end{aligned}$$
- The borders correspond to the equations $x_1 = 0, x_2 = 0, x_3 = 0, x_4 = 0$.
- Start with $(x_1, x_2, x_3, x_4) = (0, 0, 1, 3)$ (the origin in the two-dimensional case).

Knowledge

In case of a bounded optimum, there is always an extreme point (vertex of the feasible region) in which the optimum is attained. In an extreme point the number of variables with a positive value is at most equal to the number of equations.

Improving current solution (1)

Problem: maximize z subject to the constraints

$$\begin{aligned} z - 2x_1 - x_2 &= 0 \\ x_1 - x_2 + x_3 &= 1 \\ x_1 + x_2 + x_4 &= 3 \\ x_1, x_2, x_3, x_4 &\geq 0 \end{aligned}$$

Put all variables with value 0 at the right-hand-side. Denote only the equations.

$$\begin{aligned} z &= 2x_1 + x_2 \\ x_3 &= 1 - x_1 + x_2 \\ x_4 &= 3 - x_1 - x_2 \end{aligned}$$

What to do next?

Improving current solution (2)

- Increase the value of *one* variable with current value equal to 0 (x_1 or x_2).
- Increase a variable such that the value of z increases; if this is not possible, then you have found an optimum solution.
- All other variables at the right-hand-side remain equal to 0; adjust the left-hand-side variables according to the equations.

$$
\begin{aligned}
z &= 2x_1 + x_2 \\
x_3 &= 1 - x_1 + x_2 \\
x_4 &= 3 - x_1 - x_2
\end{aligned}
$$

Improving current solution (3)

- The objective equation $z = 2x_1 + x_2$ indicates that increasing x_1 improves the objective value with 2 per unit; increasing x_2 leads to a gain of 1 per unit.
- Choose (greedy) to increase x_1 (walk along the border.)
- Increase x_1 maximally until one of the other variables becomes 0 (you hit at an extreme point).

$$\begin{aligned} z &= 2x_1 + x_2 \\ x_3 &= 1 - x_1 + x_2 \\ x_4 &= 3 - x_1 - x_2 \end{aligned}$$

New point: $x_3 = 0$ and $x_1 = 1$.

Improving current solution (4)

- Reformulate the problem, such that z, x_1, x_4 get expressed in x_2 and x_3 (zero-value variables).
- Use $x_3 = 1 - x_1 + x_2$, that is, $x_1 = 1 - x_3 + x_2$; use this to rearrange the other equations.

$$\begin{array}{rcl} z & = & 2 + 3x_2 - 2x_3 \\ x_1 & = & 1 + x_2 - x_3 \\ x_4 & = & 2 - 2x_2 + x_3 \end{array}$$

Improving current solution (5)

$$\begin{aligned} z &= 2 + 3x_2 - 2x_3 \\ x_1 &= 1 + x_2 - x_3 \\ x_4 &= 2 - 2x_2 + x_3 \end{aligned}$$

- Increasing x_2 yields 3 per unit; increasing x_3 costs 2 per unit.
- Increase x_2 maximally $\Longrightarrow$ x_4 drops to 0.
- Adjust again the equations, use $x_4 = 2 - 2x_2 + x_3 \Leftrightarrow x_2 = 1 + 0.5x_3 - 0.5x_4$
- Express z, x_1, x_2 in x_3 and x_4.

$$\begin{aligned} z &= 5 - 0.5x_3 - 1.5x_4 \\ x_1 &= 2 - 0.5x_3 - 0.5x_4 \\ x_2 &= 1 + 0.5x_3 - 0.5x_4 \end{aligned}$$

Optimal solution

$(x_1, x_2, x_3, x_4) = (2, 1, 0, 0)$ with value 5.

Solution options

A linear programming problem can

- be infeasible
  - Examples:
    $\max\{6x_1 + 4x_2 | x_1 + x_2 \leq 3, 2x_1 + 2x_2 \geq 8, x_1, x_2 \geq 0\}$
    $\max\{x_1 | x_1 \geq 1\}$

- be unbounded
  - Example:
    $\max\{6x_1 + 4x_2 | x_1 + x_2 \geq 3, 2x_1 + 2x_2 \geq 8, x_1, x_2 \geq 0\}$
    for any $\lambda \geq 2$ we have that $x_1 = \lambda, x_2 = \lambda$ is feasible

- have a bounded optimum