Discrete Optimization

- A new graduate course, offered in Block 1 (September 2014).
- Teachers: Martin Zachariasen and Pawel Winter (course responsible)
- This lecture is a brief introduction and an appetizer.
- You can still sign up (eftertilmelding)

Operations Research

- Deals with the application of advanced analytical methods to help making better decisions.
- Uses mathematical modeling, statistical analysis, and mathematical optimization, to obtain optimal or near-optimal solutions to complex decision-making problems.
- Objectives: Maximize profit, performance, or yield or minimize loss, risk, or cost.

Mathematical Models

- A mathematical model is a description of a system using mathematical concepts and language.
- Example: The facility location problem consists of *n* potential facility sites and a set of *m* demand points to be served. The goal is to open a subset of facilities minimizing
  - sum of distances from each demand point to the facility it is assigned to,
  - sum of opening costs of the facilities.

Mathematical Model

- y_i, i = 1,2,...,n: 0-1 variables, y_i = 1 open facility f_i.
- c_i, i = 1,2,...,n: cost of opening facility f_i.
- x_{ij}, i = 1,2,...,n, j = 1,2, ...,m: 0-1 variables, x_{ij} =1 facility f_i serves customer j.
- d_{ij}, i = 1,2, ..., n, j = 1,2, ...,m: cost of serving customer j from facility i.

$$min \sum_{i=1}^{n} c_i y_i + \sum_{i=1}^{n} \sum_{j=1}^{m} d_{ij} x_{ij}$$

$$s.t. \sum_{i=1}^{n} x_{ij} = 1, j = 1,2,\ldots,m$$

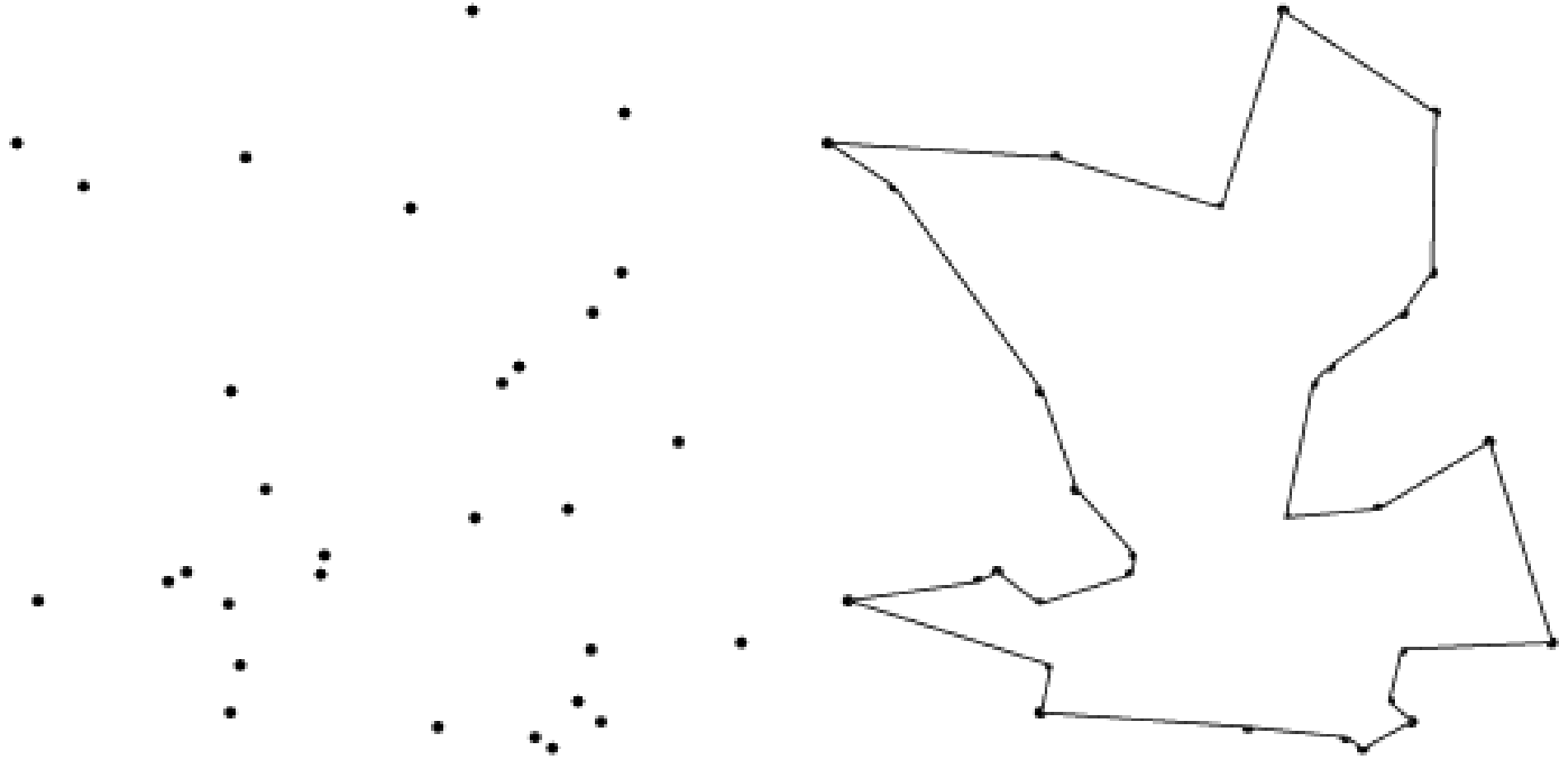
$$y_i \geq x_{ij}, i = 1,2,\ldots,n, j = 1,2,\ldots,m$$

$$y_i \text{ and } x_{ij} \text{ binary for all } i \text{ and } j$$

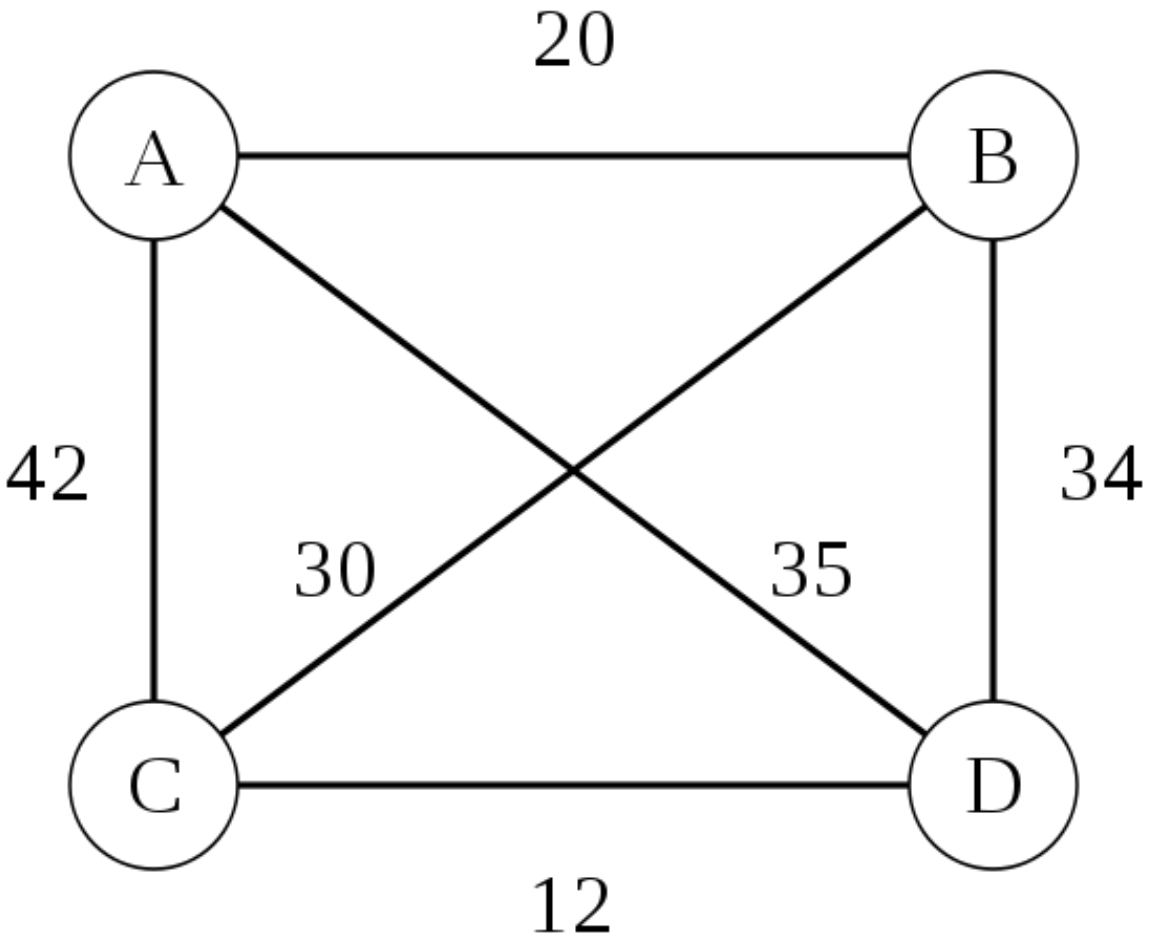
Aplications of ILP

- Vehicle routing.
- Scheduling and distribution problems.
- Production planning.
- Packing problems.
- Facility location.
- Telecommunication networks.
- And many, many other applications.

Traveling Salesman Problem

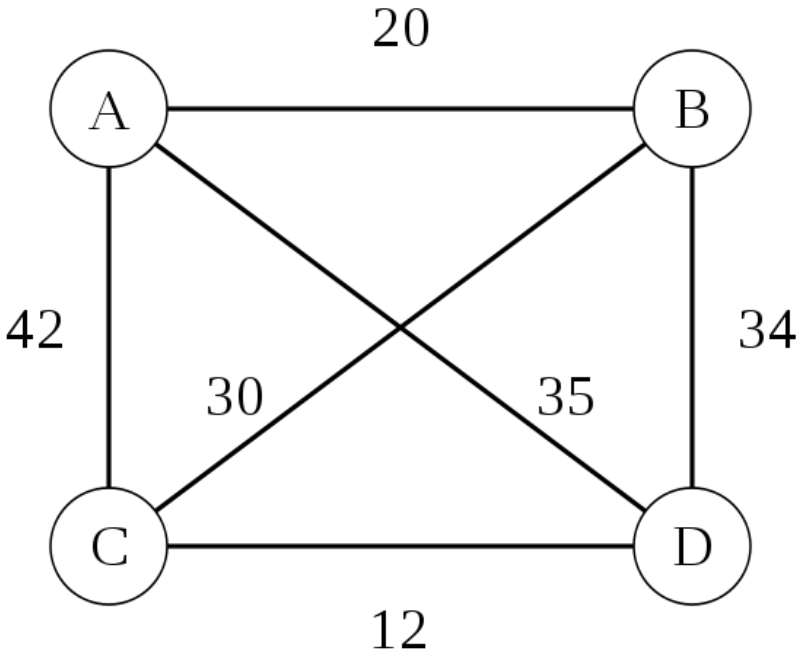


Symmetric TSP



TSP – Brute Force

- (A,B,C,D) 20+30+12+35=97
- (A,B,D,C) 20+34+12+42=108
- (A,C,B,D) 42+30+34+35=151
- (A,C,D,B) 42+12+34+20=108
- (A,D,B,C) 35+34+30+42=151
- (A,D,C,B) 35+12+30+20=97



- Number of possible tours in TSP: (n-1)!/2

TSP – Dynamic Programming

$$S \subseteq V \setminus \{1\}, t \in S$$

$$c(S,t): cost\ of\ shortest\ 1-t\ path\ through\ S$$

$$c(\{t\},t) = w(1,t)$$

$$c(S,t) = min_{m \in S \setminus \{t\}} \{c(S \setminus \{t\}, m) + w(m,t)\}$$

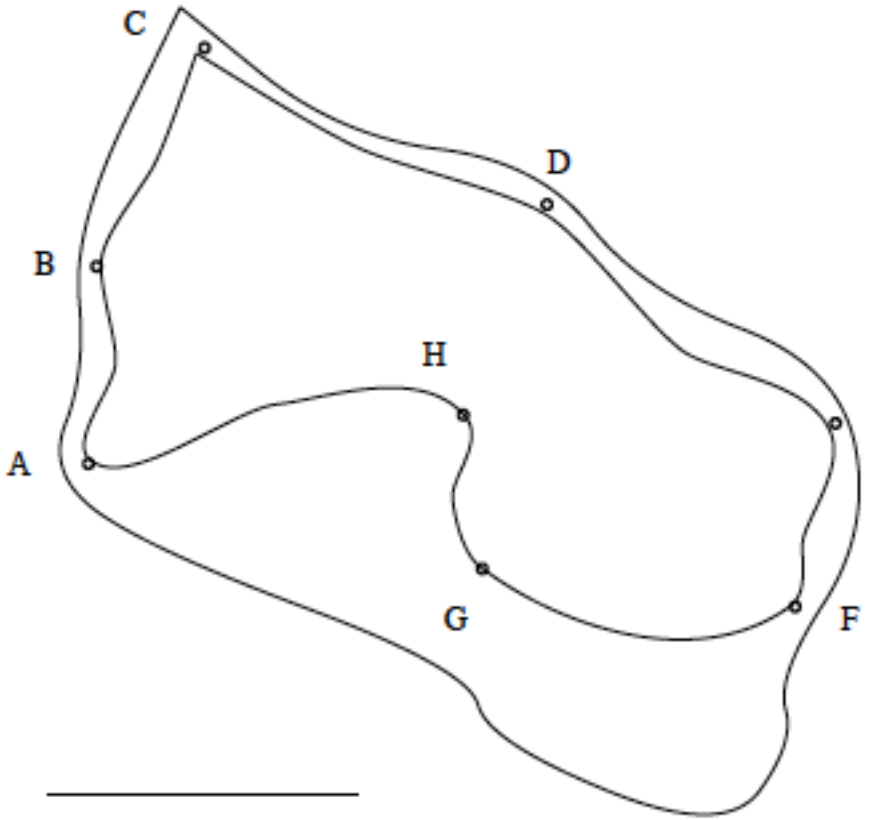
$$C = min_{l \in V \setminus \{1\}} \{c(V \setminus \{l\}, l) + w(l,1)\}$$

$$O(n^2 2^n)\ algorithm$$

Branch and Bound

- Find the minimum value of a function *f* where the variable vector *x* ranges over some set *S* of feasible solutions.
- *S* could for example be the set of traveling salesman tours through all vertices.
- Branching: Partition of *S* into 2 or more subsets of feasible solutions.
- Bounding: Lower and upper bounds for the minimal value of *f* within a given subset of *S*.

TSP of Bornholm



	A	B	C	D	E	F	G	H
A	0	11	24	25	30	29	15	15
B	11	0	13	20	32	37	17	17
C	24	13	0	16	30	39	29	22
D	25	20	16	0	15	23	18	12
E	30	32	30	15	0	9	23	15
F	29	37	39	23	9	0	14	21
G	15	17	29	18	23	14	0	7
H	15	17	22	12	15	21	7	0

Optimal tour is A-B-C-D-E-F-G-H-A and has length 100

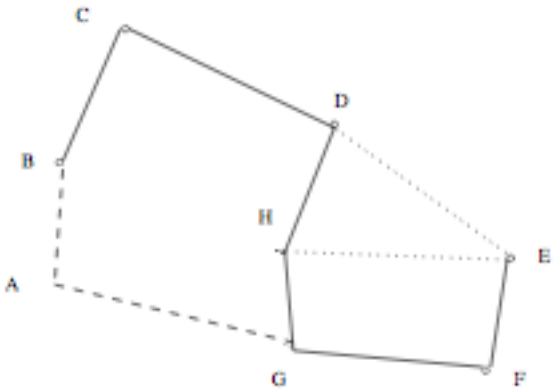
MST provides a lower bound. In this example its length is 81 = 7+9+11+12+13+14+15

Source: http://www2.imm.dtu.dk/courses/02713/NewLectures/bbtsp4.pdf

1-Tree

- Pick any vertex *v* of *G*.
- Remove *v* and its incident edges from *G*.
- Determine MST *T* in the resulting graph.
- To obtain 1-tree, add to *T* two shortest edges incident with *v*.

	A	B	C	D	E	F	G	H
A	0	11	24	25	30	29	15	15
B	11	0	13	20	32	37	17	17
C	24	13	0	16	30	39	29	22
D	25	20	16	0	15	23	18	12
E	30	32	30	15	0	9	23	15
F	29	37	39	23	9	0	14	21
G	15	17	29	18	23	14	0	7
H	15	17	22	12	15	21	7	0



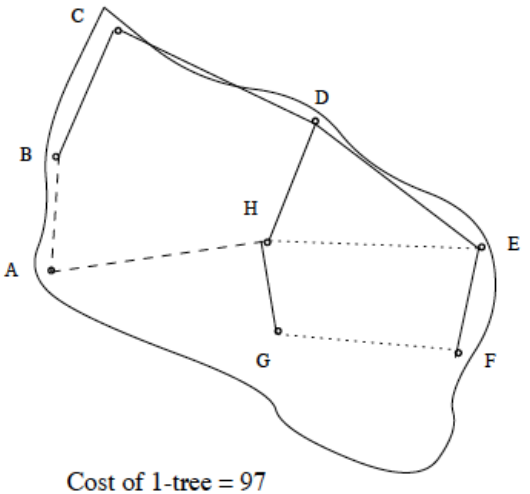
Cost of 1-tree = 97

1-Tree is a Lower Bound for TSP

- Consider an optimal tour.
- It has two edges incident with *v*.
- The tour without these two edges is a spanning tree in the reduced graph.
- This tree is not shorter than the MST.
- The removed pair of edges is not shorter than the pair of shortest edges.
- 1-tree is therefore a lower bound.

Improving 1-Tree Bound

- Change weights without affecting costs of Hamiltonian tours. While tours are unaffected, 1-tree bound may increase.
- Given 1-tree, let $p(v) = \deg(v)-2$ for all vertices.
- Let $c'(u,v) = c(u,v) + p(u) + p(v)$ for all edges.

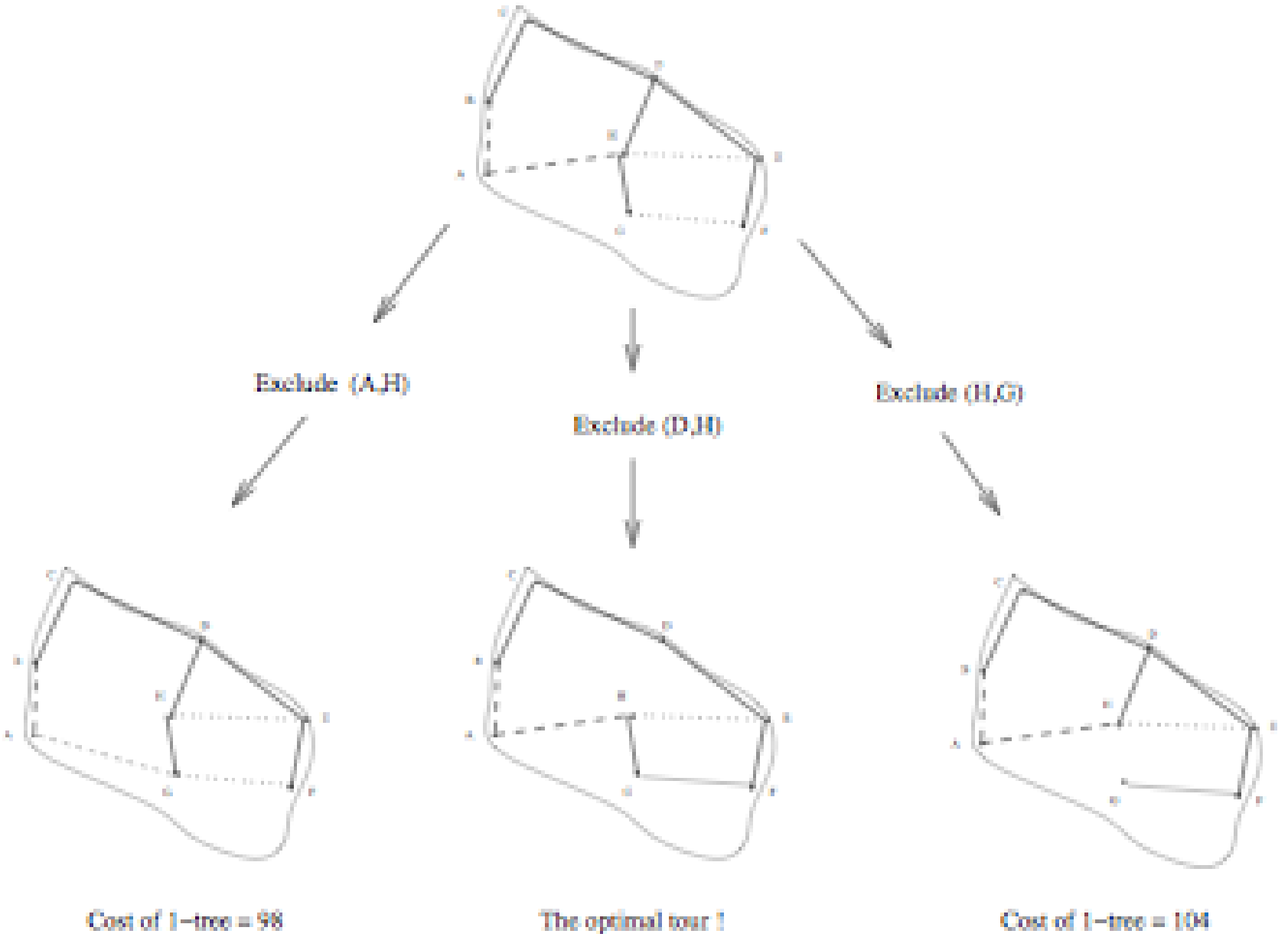


Cost of 1-tree = 97

Modified distance matrix:

	A	B	C	D	E	F	G	H
A	0	11	24	25	29	29	16	15
B	11	0	13	20	31	37	18	17
C	24	13	0	16	29	39	30	22
D	25	20	16	0	14	23	19	12
E	29	31	29	14	0	8	23	14
F	29	37	39	23	8	0	15	21
G	16	18	30	19	23	15	0	8
H	15	17	22	12	14	21	8	0

Branching on Bornholm



A-TSP – MTZ Formulation

$$x_{ij} = \begin{cases} 1 & \text{if edge } (i,\ j) \text{ is in the optimal tour} \\ 0 & \text{otherwise} \end{cases}$$

$$\min \sum_{i=1}^{n} \sum_{j=1,\ j \neq i}^{n} c_{ij}\, x_{ij}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} x_{ij} = 1, \quad j = 1,\ 2, \ldots, n \qquad \sum_{j=1}^{n} x_{ij} = 1, \quad i = 1,\ 2, \ldots, n$$

$$u_i - u_j + (n-1)\, x_{ij} \leqslant n-2, \quad 2 \leqslant i \neq j \leqslant n$$

$$x_{ij} \in \{0, 1\},\ i,\ j = 1, 2, \ldots, n, \qquad 2 \leq u_i \leq n,\ i = 2, 3, \ldots, n$$

u_i is the order of the node i in the tour.

Correctness

- Assignment constraints ensure that the solution consists of cycles.
- Suppose that optimal solution contains more than one cycle. Consider a cycle not going through vertex 1.
- Sum up left sides and right sides of k u-contraints corresponding to the edges on this cycle.
- u-terms cancel each other and we are left with

$$(n\text{-}1)k \leq (n\text{-}2)k$$

- a contradiction.
- Hence, optimal solution consists of one cycle.

Correctness

- We also need to show that for every Hamiltonian cycle, there are feasible *u*-values.
- Assume that H-cycles always start at vertex 1.
- Let t_i denote the step at which this tour visits vertex *i*. Let $u_i = t_i$ for all $i = 2,3,\ldots,n$.
- Let (i,j), $i \neq j$, $i \neq 1$, $j \neq 1$
- $x_{ij} = 0$: $u_i - u_j + (n - 1)0 = t_i - t_j \leq n-2$
- $x_{ij} = 1$: $u_i - u_j + (n - 1)1 = t_i - (t_i+1) + n-1 = n-2$

A-TSP – Subtour Formulation

$$x_{ij} = \begin{cases} 1 & \text{if edge } (i, j) \text{ is in the optimal tour} \\ 0 & \text{otherwise} \end{cases}$$

$$\min \sum_{i=1}^{n} \sum_{j=1,\, j \neq i}^{n} c_{ij}\, x_{ij}$$

s.t. $\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, 2, \ldots, n$ $\qquad$ $\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, 2, \ldots, n$

$$\sum_{i \in S,\, j \notin S} x_{ij} \geq 1, \quad S \subset V,\, S \neq \emptyset$$

$$x_{ij} \in \{0, 1\}, \quad i, j = 1, 2, \ldots, n$$

How to obtain a lower bound?

- Remove subtour constraints. Perhaps the relaxed problem is easy to solve and gives a good lower bound.
- Lagrangian relaxation: Move complicating constraints into the objective function. Solve and “panish” violated constraints in the objective function by appropriate change of their multipliers. Repeat. Lower bound at each iteration.
- Linear relaxation: Remove integrality constraints. Solve the LP problem.

Assignment Problem

- Given *n* x *n* cost matrix *C*, find *n* entries so that each entry is in its own row and its own column and the sum of the costs of the entries is minimized.
- Theorem: If a number is added to or subtracted from all entries of a row or a column of *C*, then on optimal assignment for the resulting cost matrix *C′* is also an optimal assignment for *C*.

Hungarian Algorithm

- Step 1. Subtract the smallest entry in each row from all row entries.
- Step 2. Subtract the smallest entry in each column from all column entries.
- Step 3. Draw lines through appropriate rows and columns so that all zeros are covered and the minimum number of such lines is used.
- Step 4. If the minimum number of covering lines is *n*, an optimal assignment of zeros is found. STOP.
- Step 5. Determine the smallest entry not covered by any line. Subtract this entry from each uncovered row, and then add it to each covered column. Return to Step 3.

Hungarian Algorithm - Example

90	75	75	80
35	85	55	65
125	95	90	105
45	110	95	115

Subtract min in each row

15	0	0	5
0	50	20	30
35	5	0	15
0	65	50	70

Subtract min from each column

Find min cover of 0s

15	0	0	0
0	50	20	25
35	5	0	10
0	65	50	65

Subtract 5 from each uncovered row

15	0	0	0
-5	45	15	20
30	0	-5	5
-5	60	45	60

Add 5 to each covered column

Find min cover of 0s

20	0	5	0
0	45	20	20
35	0	0	5
0	60	50	60

Subtract 20 from each uncovered row. Add 20 to each covered column.

40	0	5	0
0	25	0	0
55	0	0	5
0	40	30	40

General LP Problem

$$\begin{aligned} min \quad & \sum_{j=1}^{n} c_j x_j \\ s.t. \quad & m \text{ linear constraints} \end{aligned}$$

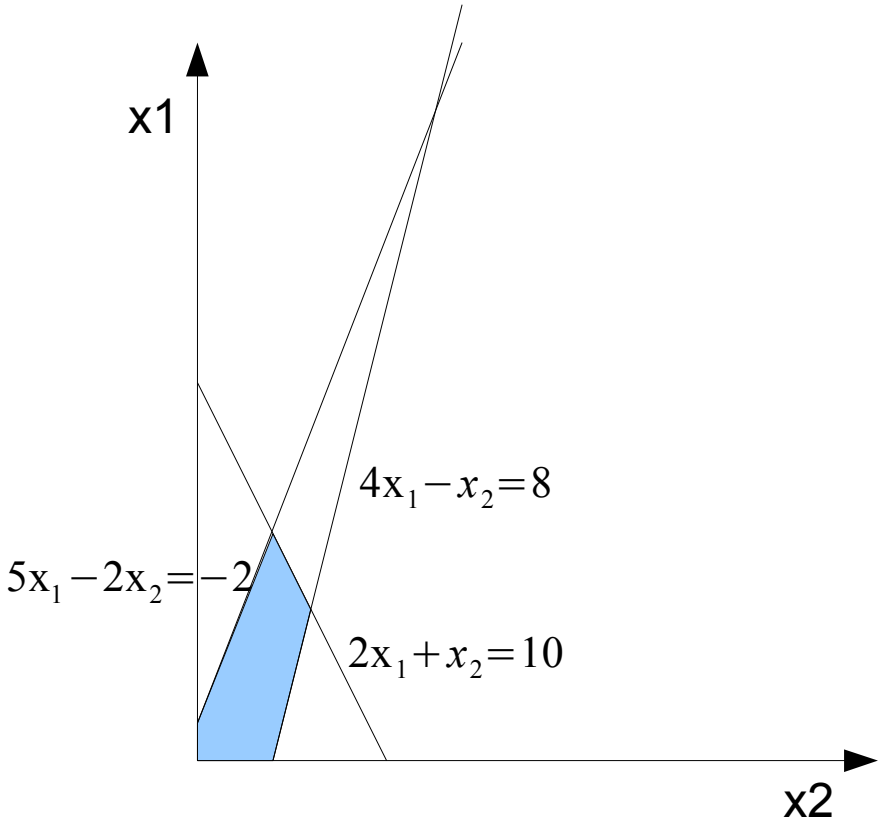
- Minimization or maximization of a **linear** objective function with *n* real-valued variables.
- An optimal solution must satisfy *m* **linear** constraints (inequalities or equalities).
- Strict inequalities are not allowed.
- ”programming” in ”linear programming” does not refer to any code. It was chosen before computer programming was born.

Applications of LP

- Scheduling problems: airline wishes to schedule its flight crews on all flights while using as few crew members as possible.
- Manufacturing problems: How much of each type of product should be produced subject to technical and financial constraints.
- Location problems: Locating drills to maximize the amount of oil that will be extracted under given budget constraints.
- Many network and graph problems can be formulated as LP; dimensionig telecommunication and distribution networks.
- Allocation of financial assets to maximize profit or minimize risk.
- Integer linear programming problems.

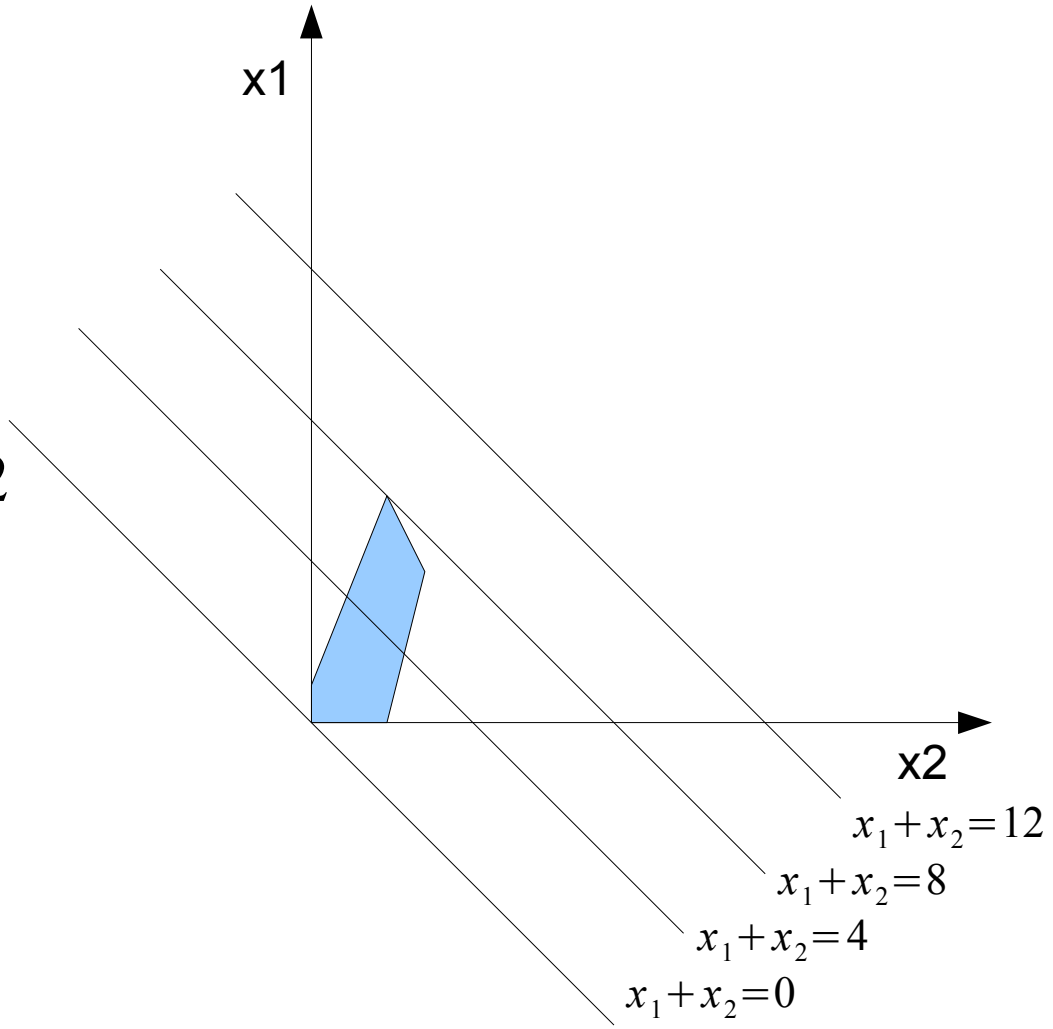
Geometric Interpretation

$$
\begin{array}{llrlr}
max & x_1 & + & x_2 & \\
s.t. & 4x_1 & - & x_2 & \leq 8 \\
 & 2x_1 & + & x_2 & \leq 10 \\
 & 5x_1 & - & 2x_2 & \geq -2 \\
 & x_1, & & x_2 & \geq 0
\end{array}
$$



Geometric Interpretation

$$
\begin{array}{lrcrcr}
max & x_1 & + & x_2 & & \\
s.t. & 4x_1 & - & x_2 & \leq & 8 \\
 & 2x_1 & + & x_2 & \leq & 10 \\
 & 5x_1 & - & 2x_2 & \geq & -2 \\
 & x_1, & & x_2 & \geq & 0
\end{array}
$$



General Idea Behind SIMPLEX Algorithm

- SIMPLEX starts with a feasible solution corresponding to some vertex of the simplex. We will show how to find such a vertex (or decide that the feasible region is empty).
- SIMPLEX keeps ”jumping” from a vertex of the simplex to a new vertex if the new vertex offers a feasible solution that is better (or at least not worse). We will show how SIMPLEX ”jumps”. We will show how to avoid ”cycling” when SIMPLEX jumps through feasible solutions with the same objective value.
- When no more ”jumps” are possible, we will show that SIMPLEX is in an optimal vertex (or the LP is unbounded).
- We will show that the number of jumps is finite (at most equal to the number of simplex vertices).

LP in Standard Form

- *n* variables, *m* constraints

$$
\begin{aligned}
max \quad & \sum_{j=1}^{n} c_j x_j & & & & \\
s.t. \quad & \sum_{j=1}^{n} a_{ij} x_j & \leq & b_i & for & \quad i = 1, 2, \ldots, m \\
& x_j & \geq & 0 & for & \quad j = 1, 2, \ldots, n
\end{aligned}
$$

Slack Variables (Overskudsvariable)

- Consider one of the constraints, for example

$$2x_1 + 3x_2 + x_3 \leq 5$$

- For every feasible solution x_1, x_2, x_3 , the value of the left-hand side is at most the value of the right-hand side.
- Often there can be a **slack** between these two values.
- Denote the slack by x_4.
- By requiring x_4 >= 0, the inequality can be replaced by

$$2x_1 + 3x_2 + x_3 + x_4 = 5$$

Slack Variables

$$
\begin{aligned}
max \quad & \sum_{j=1}^{n} c_j x_j \\
s.t. \quad & \sum_{j=1}^{n} a_{ij} x_j \le b_i \quad for \quad i = 1, 2, \ldots, m \\
& x_j \ge 0 \qquad\qquad\quad for \quad j = 1, 2, \ldots, n
\end{aligned}
$$

$$
\begin{aligned}
max \quad & \sum_{j=1}^{n} c_j x_j \\
s.t. \quad & \sum_{j=1}^{n} a_{ij} x_j + x_{n+i} = b_i \quad for \quad i = 1, 2, \ldots, m \\
& x_j \ge 0 \qquad\qquad\qquad\quad for \quad j = 1, 2, \ldots, n+m
\end{aligned}
$$

LP in Slack Form

$$
\begin{aligned}
max \quad & \sum_{j=1}^{n} c_j x_j \\
s.t. \quad & \sum_{j=1}^{n} a_{ij} x_j + x_{n+i} = b_i \quad for \quad i = 1, 2, \ldots, m \\
& x_j \geq 0 \quad for \quad j = 1, 2, \ldots, n+m
\end{aligned}
$$

$$
\begin{aligned}
max \quad & z = \sum_{j=1}^{n} c_j x_j \\
s.t. \quad & x_{n+i} = b_i - \sum_{j=1}^{n} a_{ij} x_j \quad for \quad i = 1, 2, \ldots, m \\
& x_j \geq 0 \quad for \quad j = 1, 2, \ldots, n+m
\end{aligned}
$$

$$
\begin{aligned}
z &= 0 + \sum_{j=1}^{n} c_j x_j \\
x_{n+i} &= b_i - \sum_{j=1}^{n} a_{ij} x_j \quad for \quad i = 1, 2, \ldots, m
\end{aligned}
$$

Standard to Slack Form - Example

$$
\begin{array}{lrcrcrcr}
max & 2x_1 & - & 3x_2 & + & 3x_3 & & \\
s.t. & x_1 & + & x_2 & - & x_3 & \leq & 7 \\
 & -x_1 & - & x_2 & + & x_3 & \leq & -7 \\
 & x_1 & - & 2x_2 & + & 2x_3 & \leq & 4
\end{array}
$$

$$
\begin{array}{lrcrcrcrcrcr}
max & 2x_1 & - & 3x_2 & + & 3x_3 & & & & & & \\
s.t. & x_1 & + & x_2 & - & x_3 & + & x_4 & & & = & 7 \\
 & -x_1 & - & x_2 & + & x_3 & & & + & x_5 & = & -7 \\
 & x_1 & - & 2x_2 & + & 2x_3 & & & & + \; x_6 & = & 4
\end{array}
$$

$$
\begin{array}{rcrcrcrcr}
z & = & 0 & + & 2x_1 & - & 3x_2 & + & 3x_3 \\
x_4 & = & 7 & - & x_1 & - & x_2 & + & x_3 \\
x_5 & = & -7 & + & x_1 & + & x_2 & - & x_3 \\
x_6 & = & 4 & - & x_1 & + & 2x_2 & - & 2x_3
\end{array}
$$

Basic Solutions

- Any solution of LP in standard form yields a solution of LP in the corresponding slack form (with the same objective value) and vice versa.
- Setting right-hand side variables of the slack form to 0 yields a **basic solution.**
- Left-hand side variables are called **basic**. Right-hand side variables are called **nonbasic**.
- The basic variables are said to constitute a **basis.**
- Note that a basic solution does not need to be feasible.

SIMPLEX - Example

- LP problem in standard form:

$$
\begin{array}{lrcrcrcr}
max & 3x_1 & + & x_2 & + & 2x_3 & & \\
s.t. & x_1 & + & x_2 & + & 3x_3 & \leq & 30 \\
 & 2x_1 & + & 2x_2 & + & 5x_3 & \leq & 24 \\
 & 4x_1 & + & x_2 & + & 2x_3 & \leq & 36 \\
 & x_1 & , & x_2 & , & x_3 & \geq & 0
\end{array}
$$

SIMPLEX – Example Continued

- LP in slack form:

$$
\begin{aligned}
z &= 0 + 3x_1 + x_2 + 2x_3 \\
x_4 &= 30 - x_1 - x_2 - 3x_3 \\
x_5 &= 24 - 2x_1 - 2x_2 - 5x_3 \\
x_6 &= 36 - 4x_1 - x_2 - 2x_3
\end{aligned}
$$

- Set all **nonbasic** variables (right-hand side) to 0.
- Compute values of **basic** variables: x_4=30, x_5=24, x_6=36.
- Compute the objective value z (= 0).
- This gives the feasible basic solution (0,0,0,30,24,36).
- It is feasible; not always the case – we were lucky.

SIMPLEX: 1. Pivoting

- Can x_1 be increased without violating feasibility?

$$
\begin{array}{lcrcrcrcr}
z & = & 0 & + & 3x_1 & + & x_2 & + & 2x_3 \\
x_4 & = & 30 & - & x_1 & - & x_2 & - & 3x_3 \\
x_5 & = & 24 & - & 2x_1 & - & 2x_2 & - & 5x_3 \\
x_6 & = & 36 & - & 4x_1 & - & x_2 & - & 2x_3
\end{array}
$$

- If x_1 is increased to 1, then x_4=29, x_5=22, x_6=32 while z=3. (1,0,0,29,22,32) is a feasible solution.
- If x_1 is increased to 2, then x_4=28, x_5=20, x_6=28 while z=6. (2,0,0,28,20,28) is a feasible solution.
- If x_1 is increased to 3, then x_4=27, x_5=18, x_6=24 while z=9. (3,0,0,27,18,24) is a feasible solution.

SIMPLEX: 1. Pivoting

- Can x_1 be increased without violating feasibility? By how much?

$$
\begin{aligned}
z &= 0 + 3x_1 + x_2 + 2x_3 \\
x_4 &= 30 - x_1 - x_2 - 3x_3 \\
x_5 &= 24 - 2x_1 - 2x_2 - 5x_3 \\
x_6 &= 36 - 4x_1 - x_2 - 2x_3
\end{aligned}
$$

- If x_1 is increased beyond 30 then x_4 becomes negative.
- If x_1 is increased beyond 12 then x_5 becomes negative.
- If x_1 is increased beyond 9 then x_6 becomes negative.
- Constraint defining x_6 is **binding**.

SIMPLEX: 1. Pivoting

- So x_1 can be increased to 9 without losing feasibility. The feasible solution is (9,0,0,21,6,0) and $z = 27$.
- We will now rewrite the slack form to an equivalent slack form with x_1, x_4, x_5 as basic variables and with (9,0,0,21,6,0) being its feasible basic solution.
- This rewriting is called **pivoting.**
- Binding constraint defining x_6 is rewritten so that it has x_1 on its left-hand side.
- All occurences of x_1 in other constraints and in the objective function are replaced by the right-hand side of the binding constraint.

SIMPLEX: 1. Pivoting

$$
\begin{aligned}
z &= 0 + 3(9 - x_2/4 - x_3/2 - x_6/4) + x_2 + 2x_3 \\
x_4 &= 30 - (9 - x_2/4 - x_3/2 - x_6/4) - x_2 - 3x_3 \\
x_5 &= 24 - 2(9 - x_2/4 - x_3/2 - x_6/4) - 2x_2 - 5x_3 \\
x_1 &= 9 - x_2/4 - x_3/2 - x_6/4
\end{aligned}
$$

$$
\begin{aligned}
z &= 27 + x_2/4 + x_3/2 - 3x_6/4 \\
x_4 &= 21 - 3x_2/4 - 5x_3/2 + x_6/4 \\
x_5 &= 6 - 3x_2/2 - 4x_3 + x_6/2 \\
x_1 &= 9 - x_2/4 - x_3/2 - x_6/4
\end{aligned}
$$

New basic variables: x_1=9, x_4=21, x_5=6

New objective value z = 27

New feasible basic solution: (9,0,0,21,6,0)

SIMPLEX: 2. Pivoting

- Can x_3 be increased without violating feasibility? By how much?

$$
\begin{aligned}
z &= 27 + x_2/4 + x_3/2 - 3x_6/4 \\
x_4 &= 21 - 3x_2/4 - 5x_3/2 + x_6/4 \\
x_5 &= 6 - 3x_2/2 - 4x_3 + x_6/2 \\
x_1 &= 9 - x_2/4 - x_3/2 - x_6/4
\end{aligned}
$$

- If x_3 is increased beyond 42/5 then $x_4 < 0$.
- If x_3 is increased beyond 3/2 then $x_5 < 0$.
- If x_3 is increased beyond 18 then $x_1 < 0$.
- Constraint defining x_5 is binding.

SIMPLEX: 2. Pivoting

$$
\begin{aligned}
z &= 27 + x_2/4 + \frac{1}{2}(3/2 - 3x_2/8 + x_6/8 - x_5/4) - 3x_6/4 \\
x_4 &= 21 - 3x_2/4 - \frac{5}{2}(3/2 - 3x_2/8 + x_6/8 - x_5/4) + x_6/4 \\
x_3 &= 3/2 - 3x_2/8 - x_5/4 + x_6/8 \\
x_1 &= 9 - x_2/4 - \frac{1}{2}(3/2 - 3x_2/8 + x_6/8 - x_5/4) - x_6/4
\end{aligned}
$$

$$
\begin{aligned}
z &= 111/4 + x_2/16 - x_5/8 - 11x_6/16 \\
x_4 &= 69/4 + 3x_2/16 + 5x_5/8 - x_6/16 \\
x_3 &= 3/2 - 3x_2/8 - x_5/4 + x_6/8 \\
x_1 &= 33/4 - x_2/16 + x_5/8 - 5x_6/16
\end{aligned}
$$

New basic variables: x_1=33/4, x_3=3/2, x_4=69/4

New objective value z = 27.75

New feasible basic solution: (33/4, 0, 3/2, 69/4, 0, 0)

SIMPLEX: 3. Pivoting

- Can x_2 be increased without violating feasibility? By how much?

$$
\begin{aligned}
z &= 111/4 + x_2/16 - x_5/8 - 11x_6/16 \\
x_4 &= 69/4 + 3x_2/16 + 5x_5/8 - x_6/16 \\
x_3 &= 3/2 - 3x_2/8 - x_5/4 + x_6/8 \\
x_1 &= 33/4 - x_2/16 + x_5/8 - 5x_6/16
\end{aligned}
$$

- If x_2 is increased then x_4 also increases.
- If x_2 is increased beyond 4 then $x_3 < 0$.
- If x_2 is increased beyond 132 then $x_1 < 0$.
- Constraint defining x_3 is binding.

SIMPLEX: 3.Pivoting

$$
\begin{aligned}
z &= 111/4 + \frac{1}{16}(4 - 8x_3/3 - 2x_5/3 + x_6/3) - x_5/8 - 11x_6/16 \\
x_4 &= 69/4 + \frac{1}{16}(4 - 8x_3/3 - 2x_5/3 + x_6/3) + 5x_5/8 - x_6/16 \\
x_2 &= 4 - 8x_3/3 - 2x_5/3 + x_6/3 \\
x_1 &= 33/4 - \frac{1}{16}(4 - 8x_3/3 - 2x_5/3 + x_6/3) + x_5/8 - 5x_6/16
\end{aligned}
$$

$$
\begin{aligned}
z &= 28 - x_3/6 - x_5/6 - 2x_6/3 \\
x_4 &= 18 - x_3/2 + x_5/2 + 0x_6 \\
x_2 &= 4 - 8x_3/3 - 2x_5/3 + x_6/3 \\
x_1 &= 8 + x_3/6 + x_5/6 - x_6/3
\end{aligned}
$$

New basic variables: x_1=8, x_2=4, x_4=18

New objective value z = 28

New feasible basic solution: (8, 4, 0, 18, 0, 0) is optimal

SIMPLEX

- SIMPLEX starts with a slack form corresponding to some feasible basic solution and iterates:
  - Select a nonbasic variable x_e with $c_e > 0$. If no such x_e exists, SIMPLEX terminates. We will show that the feasible basic solution is optimal.
  - Select a basic variable x_l that most severely limits the nonbasic variable x_e. Ties are broken arbitrarily. We will show that LP is unbounded if no such x_l exists.
  - Pivot.

SIMPLEX – Open Issues

- How to decide that LP is feasible?
- What to do if the initial basic solution is infeasible?
- How to select entering and leaving variables?
- How to decide that LP is unbounded?
- Does SIMPLEX terminate?
- Does it terminate with an optimal solution?

Cutting Planes

- Consider an LP- relaxation (assuming that it has an optimum solution).
- Test the LP-optimum for being integer.
- If not, there exists a **cut** (linear inequality) that separates the LP-optimum from the feasible integer solutions.
- Finding such a cut is the **separation problem.**
- Add the cut to the LP-relaxation. The LP-solution is no longer feasible. Repeat until LP-optmum is integer.

Gomory Cuts

- Consider an IP problem:

$$max \sum_{j=1}^{n} c_j x_j$$
$$s.t. \sum_{j=1}^{n} a_{ij} x_j = b_i, i = 1, 2, \ldots, m$$
$$x_j \geq 0, \ integer, \ j = 1, 2, \ldots, n$$

- Drop integer constraints and solve the LP using the simplex.
- The constraints in the (final) slack formulation are

$$x_i = b_i - \sum_{j \in N} a_{ij} x_j, \quad i = 1, 2, \ldots, m$$

- for every basic variable x_i. If all b_i are integers, then the basic feasible solution is all integer and we have found an optimal solution to the ILP problem.
- Assume that at least one b_i is fractional.

Gomory Cuts

- Consider a fractional constraint

$$x_i = b_i - \sum_{j \in N} a_{ij} x_j$$

- By rewriting, we get

$$x_i + \sum_{j \in N} \lfloor a_{ij} \rfloor x_j - \lfloor b_i \rfloor = b_i - \lfloor b_i \rfloor - \sum_{j \in N} (a_{ij} - \lfloor a_{ij} \rfloor) x_j$$

- Can we bound RHS if all variables are integers?
- Yes, it has to be less than 1.
- What about the LHS if all variables are integer?
- It has to be an integer. Therefore

$$b_i - \lfloor b_i \rfloor - \sum_{j \in N} (a_{ij} - \lfloor a_{ij} \rfloor) x_j \leq 0$$

Gomory Cuts

- Let x_k be a new non-negative slack variable. Then

$$b_i - \lfloor b_i \rfloor - \sum_{j \in N} (a_{ij} - \lfloor a_{ij} \rfloor) x_j \leq 0$$

is equivalent to

$$b_i - \lfloor b_i \rfloor - \sum_{j \in N} (a_{ij} - \lfloor a_{ij} \rfloor) x_j + x_k = 0$$

and it does not exclude any integer feasible solution.

- By rewriting, we get

$$x_k = \lfloor b_i \rfloor - b_i - \sum_{j \in N} (\lfloor a_{ij} \rfloor - a_{ij}) x_j$$

- What if we plug the feasible basic solution into this constraint? All non-basic variables x_j are 0. Hence x_k is negative. The basic solution for the LP with the added constraint is not feasible.

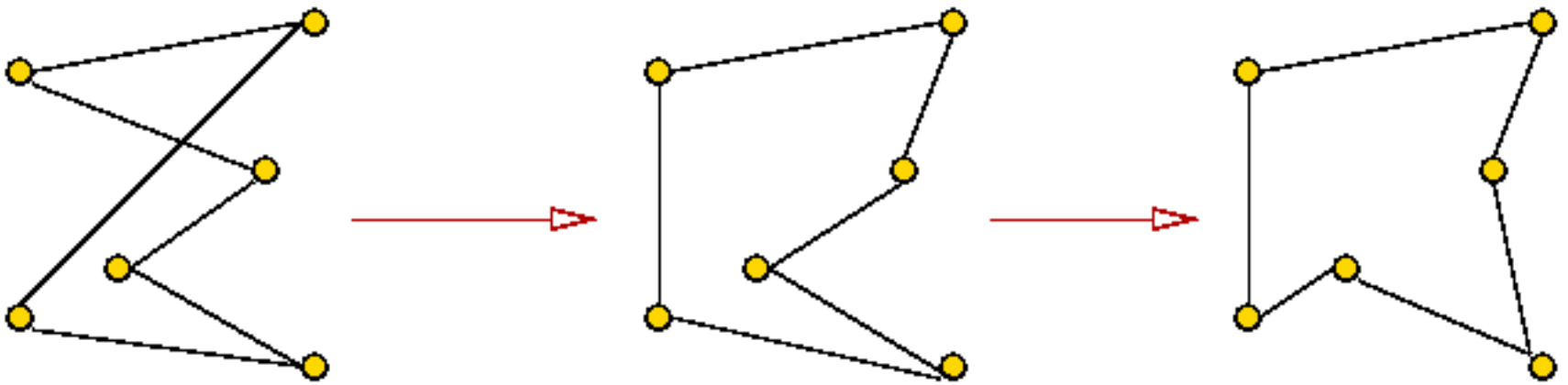
Branch and Cut

- Add cuts during the branch-and-bound.
- Method of choice to solve symmetric TSP. Problems with up to 85.900 vertices have been solved to optimality.

(Meta)Heuristics

- Hill climbing (maximization) – look for better solutions in the neighborhood of current solution.
- Simulated annealing – accept worse solution with appropriately decreasing probability
- Reactive search optimization – machine learning methods are used to adjust local search.
- Ant colony optimization – identification of “good” paths in the solutions space causes more and more “ants” to search along such paths.
- Bee colony optimization – identification of “good” regions in the solution space causes more and more “bees” to search in these regions.

2-Opt for TSP



Christofides Approximation