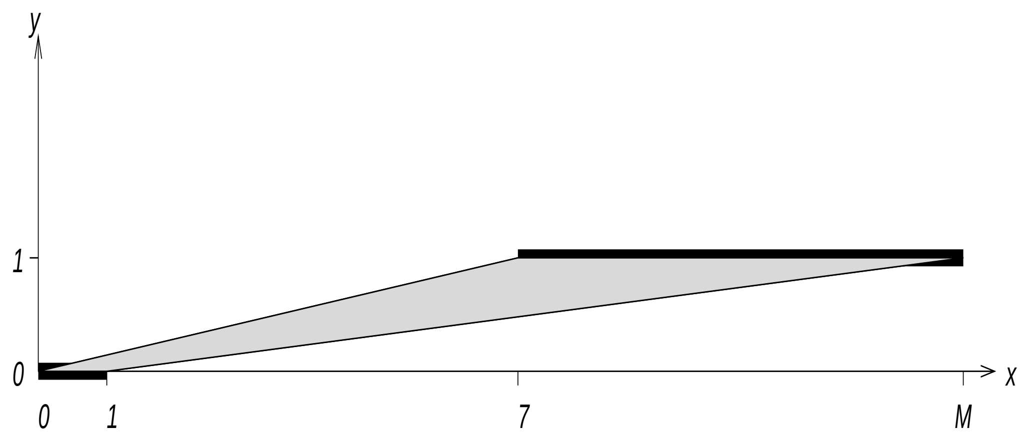
Lecture 10: Integer programming

When are integer models needed?

- Products or raw materials are indivisible
- Logical constraints: “if A then B”; “A or B”
- Fixed costs
- Combinatorics (sequencing, allocation)
- On/off-decision to buy, invest, hire, generate electricity, ...

Either $0 \le x \le 1$ or $x \ge 7$



Let $M \gg 1$: $\quad x \le 1 + My,\ x \ge 7y,\ y \in \{0, 1\}$

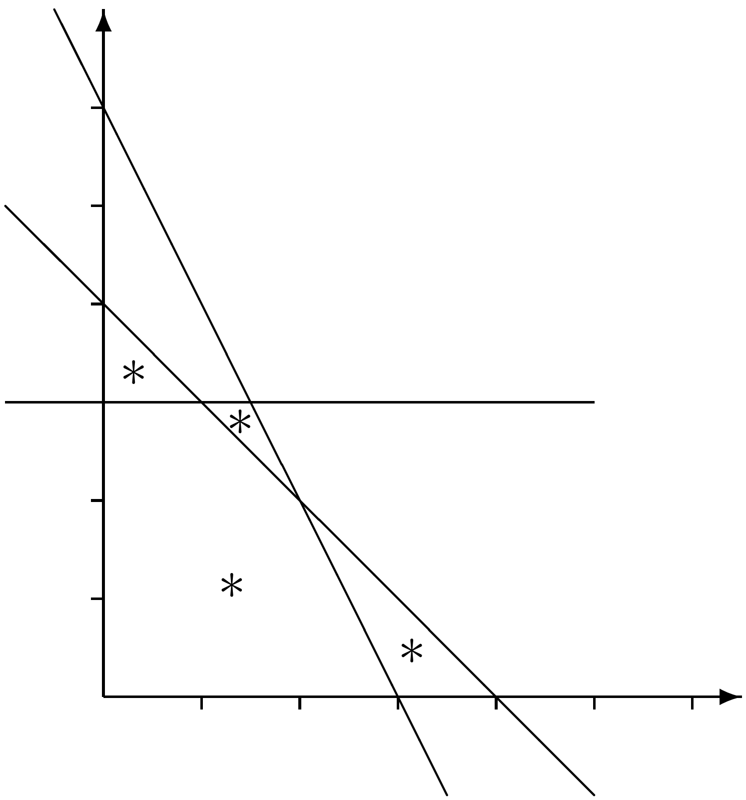
Variable x may only take the values 2, 45, 78 & 107

$$x = 2y_1 + 45y_2 + 78y_3 + 107y_4$$

$$y_1 + y_2 + y_3 + y_4 = 1$$

$$y_1, y_2, y_3, y_4 \in \{0, 1\}$$

At least 2 of 3 constraints must be fulfilled



$*=$ feasible regions

$M \geq 2$

$$\begin{aligned} x_1 + x_2 &\leq 4 && (1) \\ 2x_1 + x_2 &\leq 6 && (2) \\ x_2 &\leq 3 && (3) \\ \text{and } x_1, x_2 &\geq 0 \end{aligned}$$

$$\begin{aligned} x_1 + x_2 &\leq 4 + M(1 - y_1) && (1) \\ 2x_1 + x_2 &\leq 6 + M(1 - y_2) && (2) \\ x_2 &\leq 3 + M(1 - y_3) && (3) \\ y_1 + y_2 + y_3 &\geq 2 \\ y_1, y_2, y_3 &\in \{0, 1\} \\ \text{and } x_1, x_2 &\geq 0 \end{aligned}$$

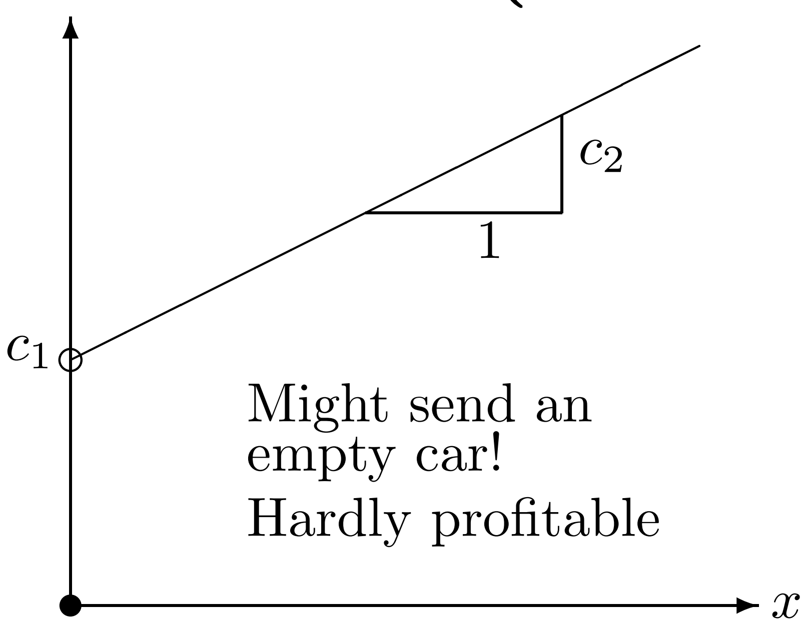
Fixed costs

$x =$ the amount of a certain product to be sent

If $x > 0$ then the initial cost c_1 (e.g. car hire) is generated

Variable cost c_2 per unit sent

$$\text{Total cost: } f(x) = \begin{cases} 0 & \text{if} \quad x = 0 \quad \boxed{\text{effect}} \\ c_1 + c_2 \cdot x & \text{if} \quad x > 0 \quad \boxed{\text{wanted!}} \end{cases}$$



Might send an empty car!

Hardly profitable

Let $M =$ car capacity

$$y = \begin{cases} 1 & \text{if } x > 0 \quad \boxed{\text{effect}} \\ 0 & \text{if } x = 0 \quad \boxed{\text{wanted!}} \end{cases}$$

$$f(x, y) = c_1 \cdot y + c_2 \cdot x$$

$$x \leq M \cdot y \qquad \boxed{\text{linear 0/1 model!}}$$

$$x \geq 0, \quad y \in \{0, 1\}$$

Other applications of integer optimization

- Facility location (new hospitals, shopping centers, etc.)
- Scheduling (on machines, personnel, projects, schools)
- Logistics (material- and warehouse control)
- Distribution (transportation of goods, buses for disabled persons)
- Production planning
- Telecommunication (network design, frequency allocation)
- VLSI design

The combinatorial explosion

Assign n persons to carry out n jobs # feasible solutions: $n!$

Assume that a feasible solution is evaluated in 10^{-9} seconds

n	2	5	8	10	100
$n!$	2	120	$4.0 \cdot 10^4$	$3.6 \cdot 10^6$	$9.3 \cdot 10^{157}$
$\lceil \text{time} \rceil$	10^{-8} s	10^{-6} s	10^{-4} s	10^{-2} s	10^{142} yrs

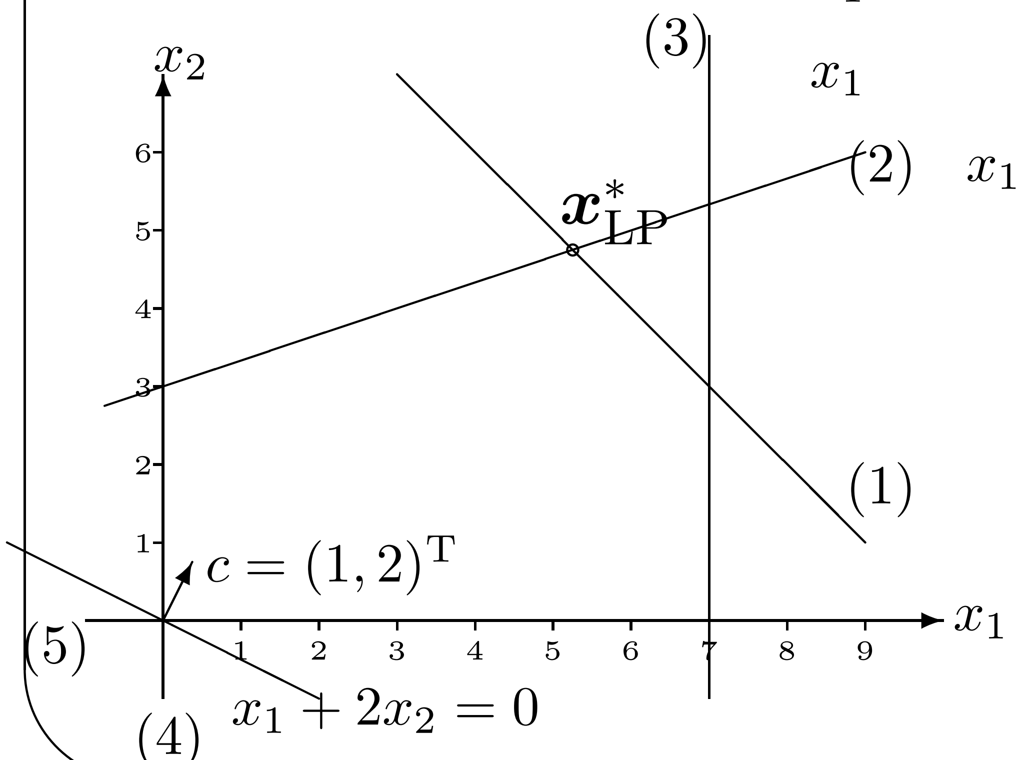
Complete enumeration of all solutions is **not** an efficient algorithm!

An algorithm exists that solves this problem in time $\mathcal{O}(n^4) \propto n^4$

n	2	5	8	10	100	1000
n^4	16	625	$4.1 \cdot 10^3$	10^4	10^8	10^{12}
$\lceil \text{time} \rceil$	10^{-7} s	10^{-6} s	10^{-5} s	$10^{-5} s$	10^{-1} s	17 min

Linear continuous optimization model

$$
\begin{array}{lcrcrcll}
\max & z_{\mathrm{LP}} = & x_1 & + & 2x_2 & & & \\
\text{s.t.} & & x_1 & + & x_2 & \leq & 10 & (1) \\
 & & -x_1 & + & 3x_2 & \leq & 9 & (2) \\
 & & x_1 & & & \leq & 7 & (3) \\
 & & & & x_1, x_2 & \geq & 0 & (4,5)
\end{array}
$$



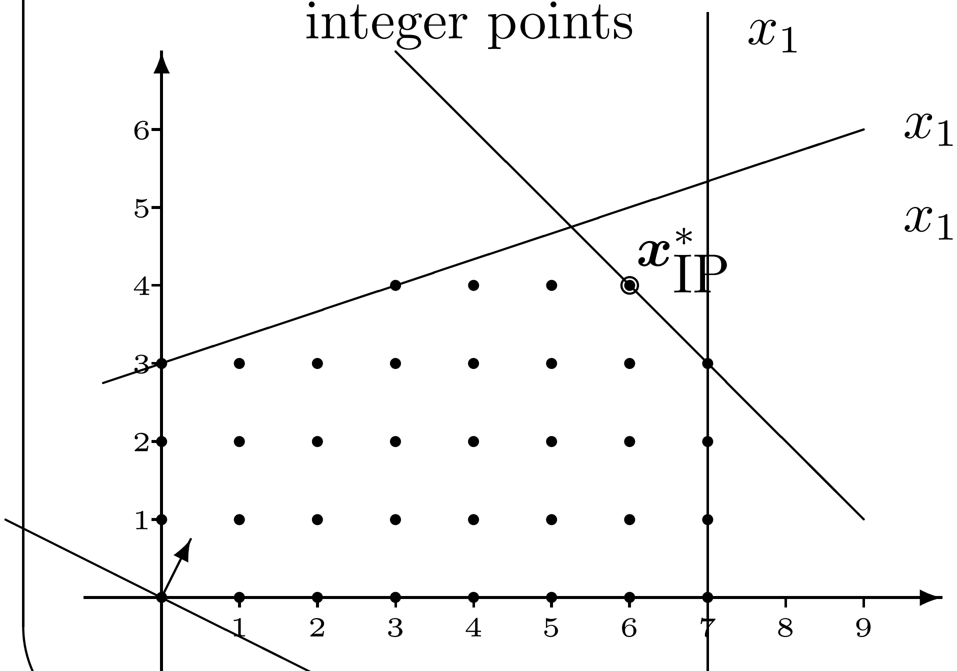
$$\boldsymbol{x}^*_{\mathrm{LP}} = \begin{pmatrix} 21/4 \\ 19/4 \end{pmatrix}$$

$$z^*_{\mathrm{LP}} = 14\tfrac{3}{4}$$

Linear integer optimization model

$$\begin{array}{rrrrrrll}
\max & z_{\mathrm{IP}} & = & x_1 & + & 2x_2 & & \\
\text{s.t.} & & & x_1 & + & x_2 & \leq & 10 \qquad (1) \\
 & & & -x_1 & + & 3x_2 & \leq & 9 \qquad (2) \\
 & & & x_1 & & & \leq & 7 \qquad (3) \\
 & & & & & x_1, x_2 & \geq & 0 \qquad (4,5) \\
 & & & & & x_1, x_2 & & \text{integer}
\end{array}$$

• = feasible integer points



$$\boldsymbol{x}^*_{\mathrm{IP}} = \begin{pmatrix} 6 \\ 4 \end{pmatrix}$$

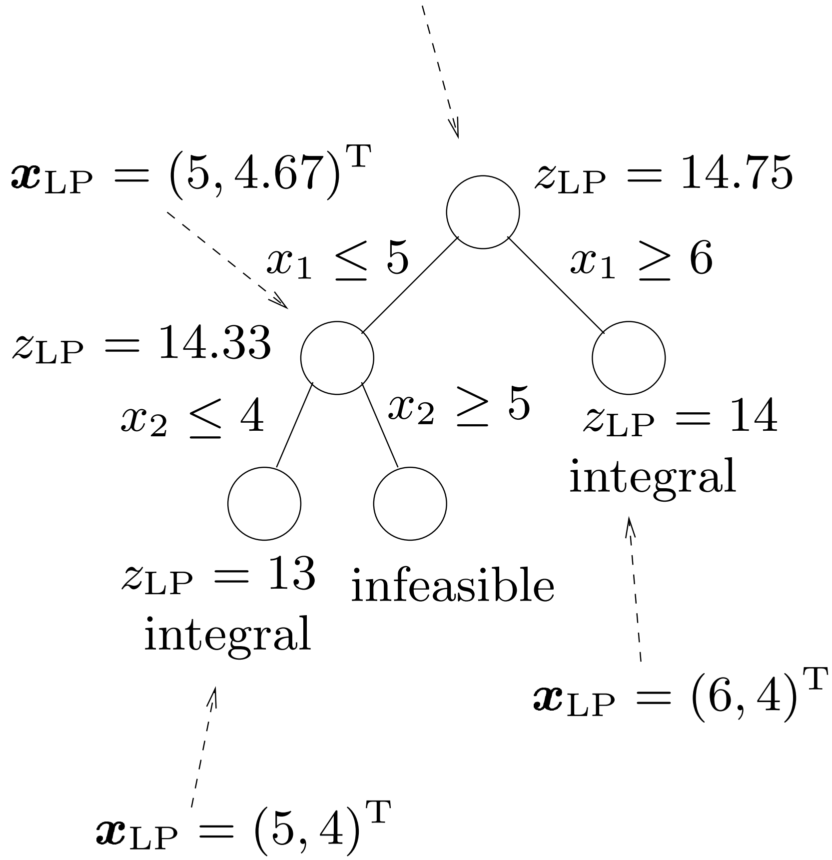
$$z^*_{\mathrm{IP}} = 14 < z^*_{\mathrm{LP}}$$

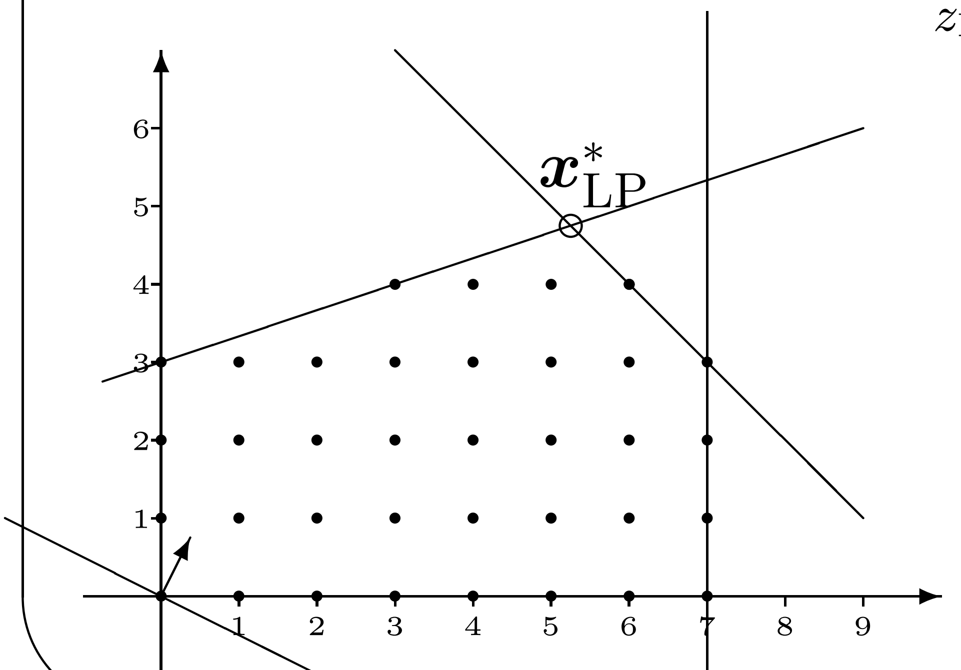
Classic methods

- *Branch–and-Bound*: relaxation plus divide–and–conquer
- *Cutting plane method*: relaxation plus generations of constraint that cut off infeasible (e.g., non-integer) points generated
- "Relaxation" can be the continuous or Lagrangian one
- *Lagrangian optimization*: Lagrangian relaxation plus multiplier optimization
- These methods are often combined (e.g., cutting planes added at nodes in B & B tree: Branch & Cut)

The branch–and–bound-algorithm

Relax integrality constraints $\Rightarrow$ linear program $\Rightarrow$ $\boldsymbol{x}^*_{\mathrm{LP}} = (5.25, 4.75)^{\mathrm{T}}$





The complexity of integer optimization, I: Aditiva

- The Aditiva LP has 62 variables and 27 linear constraints. Solution by our linux computer: 0.05 s. after 17 dual simplex pivots
- We create an integer programming (IP) variant: all producers can sell all raw materials; the suppliers have limited capacities; supplies must be bought in 100 kg batches; and there are fixed costs for transporting and for using the drying processes and the reactors
- The new problem has 168 variables (58 binary, 52 integer, 58 linear) and 131 linear constraints

- Solver uses B & B, in which to the continuous relaxation is added integer requirements on some of the binary variables that received a fractional value in the LP solution. (Note: x_j binary here $\Longrightarrow$ variable value fixed at 0 or 1)
- Solution process: after 10 minutes the solver has produced 497,000 B & B nodes and used 1,602,861 dual simplex pivots; the feasible solution found so far has not been proved to be within 0.8% from an optimal solution
- The first problem (the LP relaxation) takes only 0.06 s. and 3 dual pivots to solve

The complexity of integer optimization, II: The knapsack problem

- Knapsack problem: maximize value of a finite number of items put in a knapsack of a given capacity
- Each variable has a value and weight per unit
- AMPL model:

```
var x1..5 integer, >=0;

maximize ka:213*x[1]-1928*x[2]-11111*x[3]-2345*x[4]+9123*x[5];

subject to c1:

12223*x[1]+12224*x[2]+36674*x[3]+61119*x[4]+85569*x[5] =

89643482;
```

- Often binary; here, general integer variables

- LP relaxation trivial: sort variables in descending order of c_j/a_j; take the best one
- Result: After 10 minutes in CPLEX: 8 Million B & B nodes; no feasible solution

Cutting plane methods

- Goal: generate the convex hull of the feasible integer vectors
- Result: Can solve the IP by solving the LP relaxation over this convex hull
- Compare IP example: one extra linear constraint defines the entire convex hull! ($x_2 \leq 4$)
- Means: Relax problem (e.g., continuous relaxation); Solve. If infeasible solution, generate constraint to the relaxation that cuts off that vector but no feasible vectors. Repeat
- Constraint generation called a *separation oracle*

The Philips example—TSP solved heuristically

- Let c_{ij} denote the distance between cities i and j, with

$$\{i, j\} \subset \mathcal{N} - \text{ set of nodes}$$
$$(i, j) \in \mathcal{L} - \text{ set of links}$$

- Links (i, j) and (j, i) the same; direction plays no role
- $x_{ij} = \begin{cases} 1, & \text{if link } (i, j) \text{ is part of the TSP tour,} \\ 0, & \text{otherwise} \end{cases}$

- The Traveling Salesman Problem (TSP):

$$
\begin{array}{llll}
\text{minimize} & \displaystyle\sum_{(i,j)\in\mathcal{L}} c_{ij}x_{ij} & & \\
\text{subject to} & \displaystyle\sum_{(i,j)\in\mathcal{L}:\{i,j\}\subset S} x_{ij} \le |\mathcal{S}|-1, & \mathcal{S}\subset\mathcal{N}, & (1) \\
& \displaystyle\sum_{(i,j)\in\mathcal{L}} x_{ij} = n, & & (2) \\
& \displaystyle\sum_{i\in\mathcal{N}:(i,j)\in\mathcal{L}} x_{ij} = 2, & j\in\mathcal{N}, & (3) \\
& x_{ij}\in\{0,1\}, & (i,j)\in\mathcal{L} &
\end{array}
$$

Interpretations

- Constraint (1) implies that there can be no *sub-tours*, that is, a tour where fewer than n cities are visited (that is, if $\mathcal{S} \subset \mathcal{N}$ then there can be at most $|\mathcal{S}| - 1$ links between nodes in the set $\mathcal{S}$, where $|\mathcal{S}|$ is the cardinality–number of members of–the set $\mathcal{S}$);
- Constraint (2) implies that in total n cities must be visited;
- Constraint (3) implies that each city is connected to two others, such that we make sure to arrive from one city and leave for the next

Lagrangian relaxation

- TSP is NP-hard—no known polynomial algorithms exist
- Lagrangian relax (3) for all nodes except starting node
- Remaining problem: 1-MST—find the minimum spanning tree in the graph without the starting node and its connecting links; then, add the two cheapest links to connect the starting node
- Starting node $s \in \mathcal{N}$ and connected links assumed removed from the graph

- Objective function of the Lagrangian problem:

$$q(\boldsymbol{\lambda}) = \underset{\boldsymbol{x}}{\text{minimum}} \sum_{(i,j)\in\mathcal{L}} c_{ij}x_{ij} + \sum_{j\in\mathcal{N}} \lambda_j \Big(2 - \sum_{i\in\mathcal{N}:(i,j)\in\mathcal{L}} x_{ij} \Big)$$

$$= 2\sum_{j\in\mathcal{N}} \lambda_j + \underset{\boldsymbol{x}}{\text{minimum}} \sum_{(i,j)\in\mathcal{L}} (c_{ij} - \lambda_i - \lambda_j)x_{ij}$$

- A high (low) value of the multiplier λ_j makes node j attractive (unattractive) in the 1-MST problem, and will therefore lead to more (less) links being attached to it

- Subgradient method for updating the multipliers

- Updating step:

$$\lambda_j := \lambda_j + \alpha \left(2 - \sum_{i \in \mathcal{N}:(i,j) \in \mathcal{L}} x_{ij} \right), \qquad j \in \mathcal{N},$$

  where $\alpha > 0$ is a step length

- Update means:

  Current degree at node j :

$$\begin{cases} > 2 \Longrightarrow \lambda_j \downarrow \text{(link cost } \uparrow) \\ = 2 \Longrightarrow \lambda_j \leftrightarrow \text{(link cost constant)} \\ < 2 \Longrightarrow \lambda_j \uparrow \text{(link cost } \downarrow) \end{cases}$$

- Link cost shifted upwards (downwards) if too many (too few) links connected to node j in the 1-MST

Feasibility heuristic

- Adjusts Lagrangian solution $\boldsymbol{x}$ such that the resulting vector is feasible
- Often a good thing to do when approaching the dual optimal solution—$\boldsymbol{x}$ often then only mildly infeasible
- Identify path in 1-MST with many links; form a subgraph with the remaining nodes which is a path; connect the two
- Result: A Hamiltonian cycle (TSP tour)
- We then have both an upper bound (feasible point) and a lower bound (q) on the optimal value—a quality measure: $[f(\boldsymbol{x}) - q(\boldsymbol{\mu})]/q(\boldsymbol{\mu})$

The Philips example

- Fixed number of subgradient iterations
- Feasibility heuristic used every K iterations ($K > 1$), starting at a late subgradient iteration
- Typical example: Optimal path length in the order of 2 meters; upper and lower bounds produced concluded that the relative error in the production plan is *less than 7 %*
- Also: increase in production by some 70 %