UNIVERSITEIT GENT

FACULTY OF ENGINEERING

Column Generation Tutorial

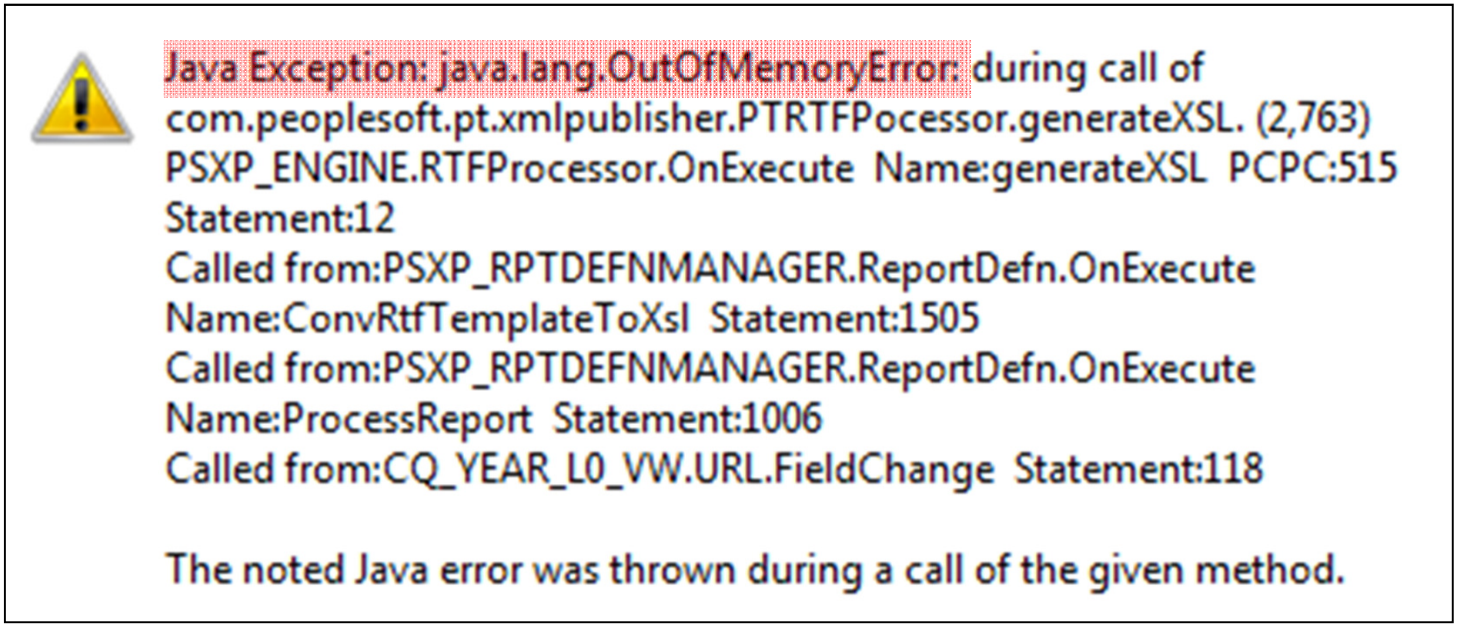
Marc De Leenheer

Ghent University – IBBT, Belgium
University of California, Davis, USA

ibbt

INTEC

Complexity of Linear Programming



- Generally applicable
  - Linear relaxation, e.g. $\{0, 1\} \rightarrow [0, 1]$
  - Lagrange relaxation
- Applicable for problems with large number of variables
  - Column generation

Linear Programming 101

- Standard form of LP model
  - Variables

$$\mathbf{x} = [x_1, x_2, \cdots, x_n]$$

  - Linear constraints

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}$$

  - Linear objective function

$$\min \mathbf{c}^T \mathbf{x}$$

- Transformation to standard form
- Additional constraints on variables
  - Integer — Integer Linear Programming or ILP
  - Mixed integer & real — MIP
  - Binary — BIP

The Simplex Method

- Feasible region = convex polytope
- **Minimum value of objective function can be found on an extreme point**
- If an extreme point is not a minimum point of the objective function, then there is an edge containing the point so that the objective function is strictly decreasing
  - Edge is finite: arrive in new extreme point
  - Edge is unbounded: LP has no solution
- Simplex algorithm always terminates, since number of vertices in polytope is finite

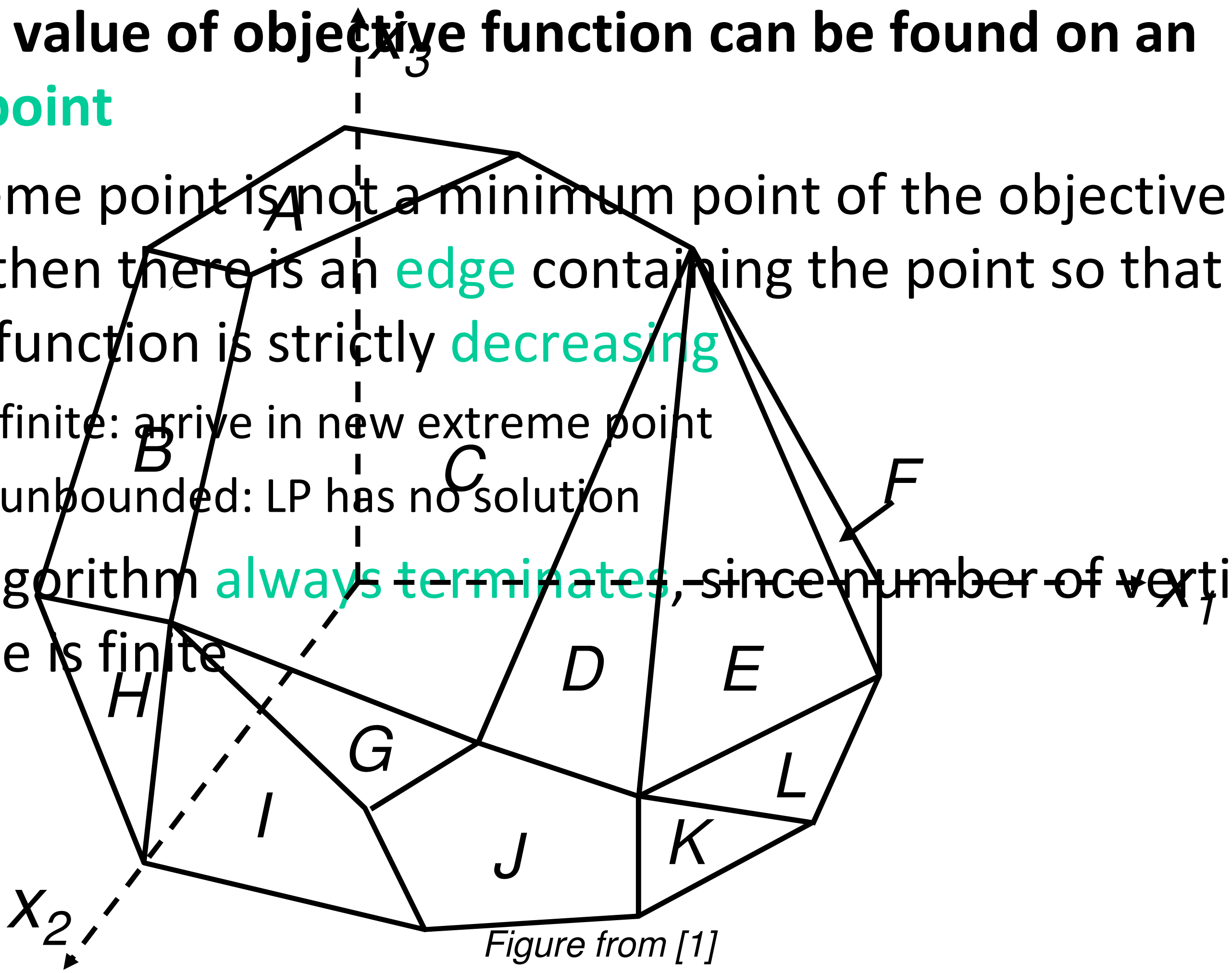
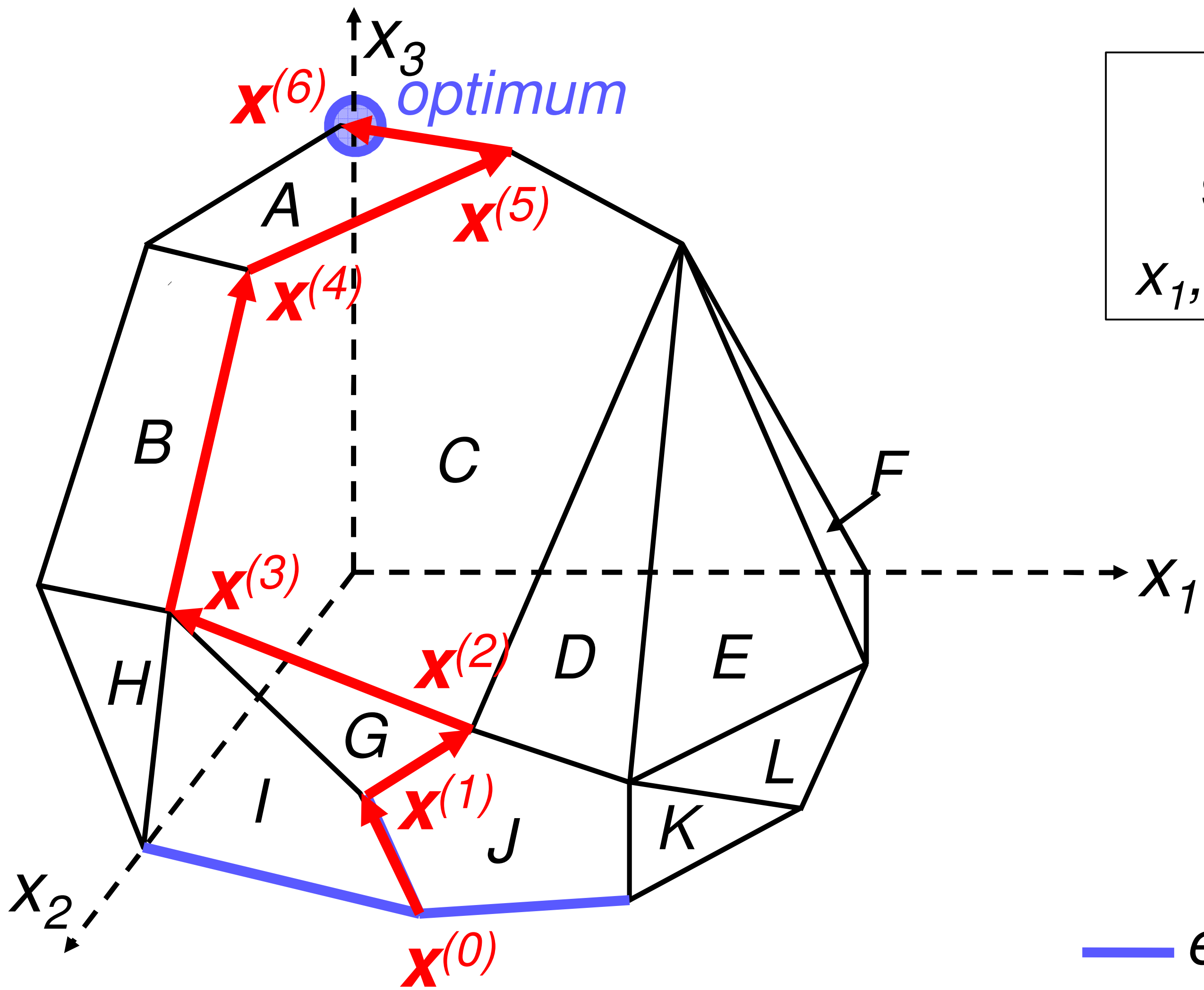


Figure from [1]

Simplex Method in Action



$$\max x_3$$
$$\text{s.t. } A\text{-}L$$
$$x_1, x_2, x_3 \geq 0$$

Improving cost

edges from $\mathbf{x}^{(0)}$

Figure from [1]

Duality

- Every LP problem has an associated dual LP problem
  - Constraint ↔ variable
  - Original is called the primal problem

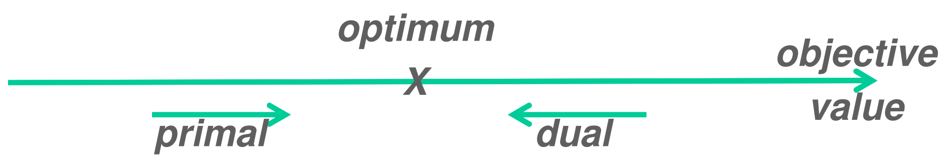
primal

$$\boxed{\begin{array}{c}\min\ \mathbf{c}^T\mathbf{x} \\ \mathbf{A}\mathbf{x} \le \mathbf{b} \quad \mathbf{x} \ge \mathbf{0}\end{array}}$$

$$\boxed{\begin{array}{c}\max\ \mathbf{b}^T\mathbf{y} \\ \mathbf{A}^T\mathbf{y} \ge \mathbf{c} \quad \mathbf{y} \ge \mathbf{0}\end{array}}$$

dual

- Weak duality theorem
  - Objective value of dual at any feasible solution ≥ objective value of primal at any feasible solution



- Strong duality theorem
  - If LP has an optimal solution, objective value of dual is the same as the primal

When using Simplex, we get the dual solution for free

Reduced Cost

- Amount by which the variable's objective function coefficient needs to improve before the variable could assume a positive value

- Estimates how much the objective function will change if you make a zero-valued variable positive

$$\boxed{\begin{array}{c}\min\ \mathbf{c}^T\mathbf{x} \\ \mathbf{A}\mathbf{x} \leq \mathbf{b} \quad \mathbf{x} \geq \mathbf{0}\end{array}}$$

Reduced cost vector σ

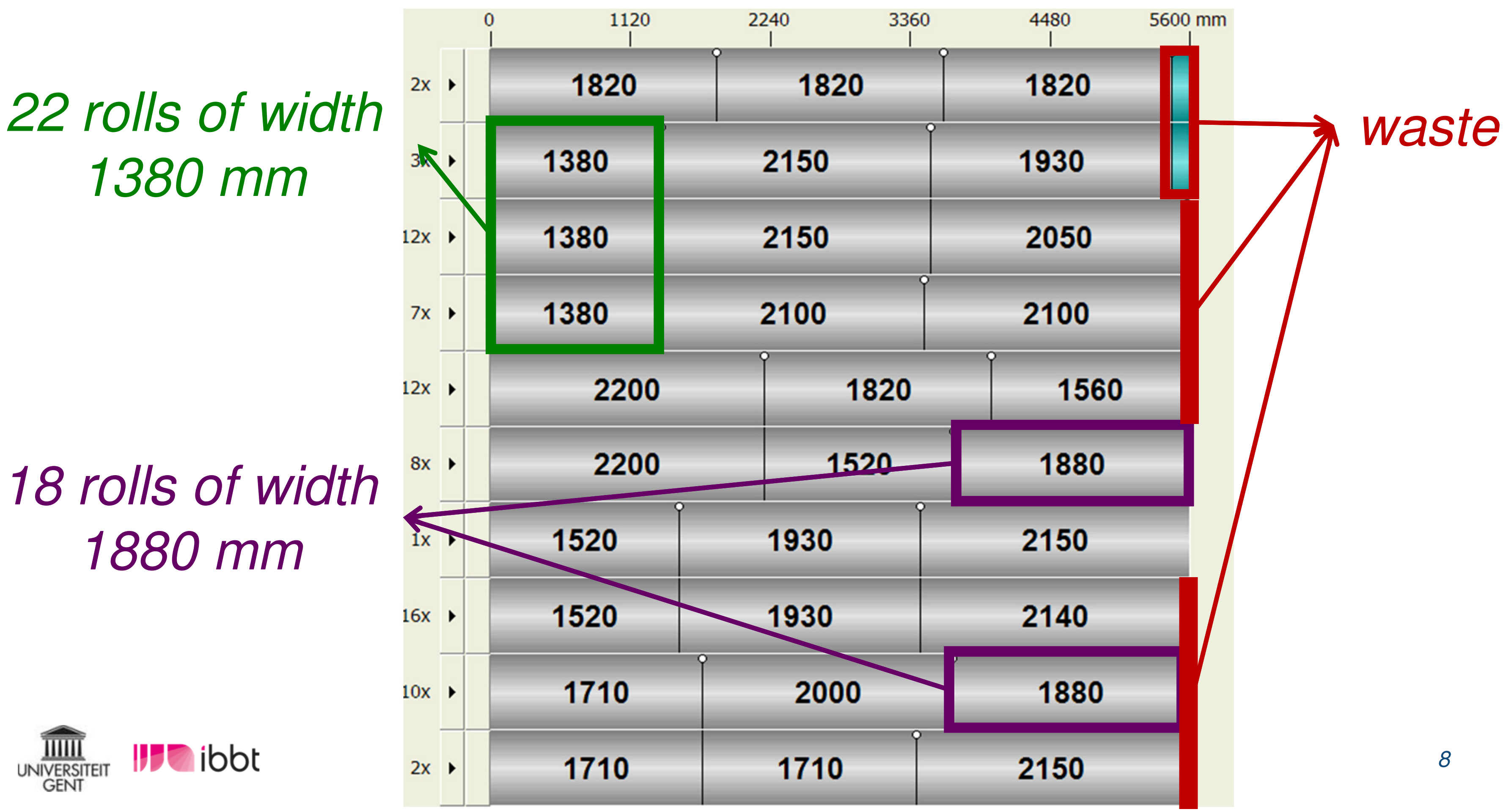
$$\boxed{\sigma = \mathbf{c} - \mathbf{A}^T\mathbf{y}}$$

dual cost vector

Column Generation: Illustrative Example

- Paper mill produces rolls of fixed width
- Customers order different number of rolls of various widths

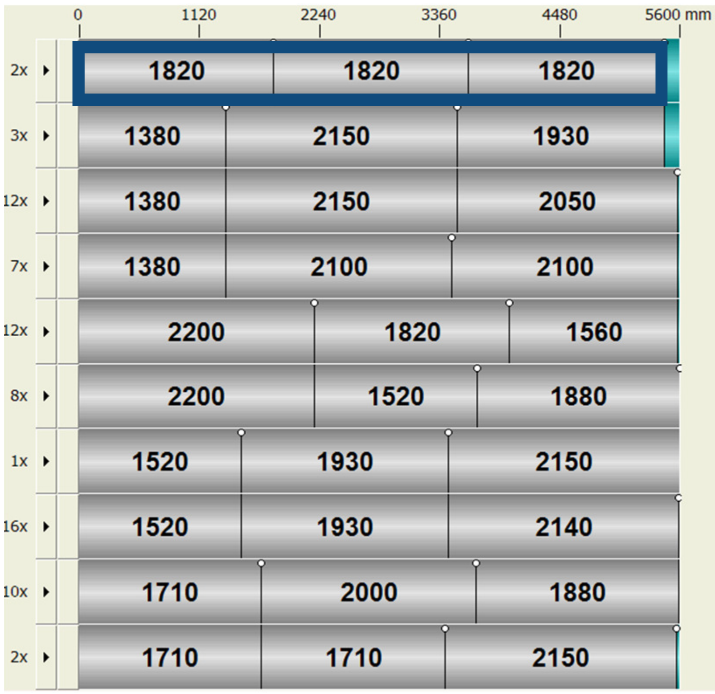
How to cut rolls and minimize waste?



Example: Cutting Stock

$W_1 = 1820$

$b_1 = 18$



$A_{11} = 3$

$A_{12} = 0$

etc.

$x_1 = 2$

etc.

- W_i width of order i
- b_i demand of order i (number of rolls)

Enumerate all possible cutting patterns

- A_{ij} number of times order i occurs in pattern j
- x_j number of times pattern j is used

$$\boxed{\begin{array}{c} \min \ \sum_j x_j \\ \text{s.t.} \ \ \sum_j A_{ij} x_j \geq b_i \\ x_j \ \text{integer} \end{array}}$$

Discussion

- Problem
  - Number of patterns increases exponentially based on number of orders
  - Leads to a large number of variables (too many, actually)

- Observation
  - Most variables in final solution equal 0, i.e. pattern is not used
  - Symmetric patterns, but can be eliminated before solving

- Solution
  - Do not generate patterns at the start, but produce them as needed

Column Generation Overview

1. Start with small set of patterns

2. Solve LP

3. Check if solution can be improved by adding a new pattern
    1. No → Optimal solution found
    2. Yes → Add pattern and go to step 2

Adding a new pattern = generating a column

(hence the name Column Generation)

Generating Columns

- Find new column with maximum reduced cost

$$\min \mathbf{c}^T\mathbf{x}$$
$$\mathbf{A}\mathbf{x} \leq \mathbf{b} \quad \mathbf{x} \geq \mathbf{0}$$

$$\downarrow$$

$$\sigma = \mathbf{c} - \mathbf{A}^T\mathbf{y}$$

$$\begin{array}{c} \min \ \sum_j x_j \\ \text{s.t.} \ \sum_j A_{ij}x_j \geq b_i \\ x_j \leq 0 \end{array}$$

$$\downarrow$$

$$\sigma_j = 1 - \sum_i A_{ji}y_i$$

If reduced cost is negative, solution can be improved

Pricing Problem

- ILP to find column with maximum reduced cost
  - Maximum = most negative

$$\boxed{\begin{array}{c} \min\ 1 - \sum_i y_j z_i \\ \text{s.t. } \sum_i W_i z_i \leq L \\ z_i \text{ integer} \end{array}}$$

- Classical knapsack problem
  - NP-hard, in theory
  - Many instances can be solved efficiently, in practice
  - E.g. use dynamic programming

Solution Diagram

$$\begin{aligned} \min\ & \textstyle\sum_j x_j \\ \text{s.t.}\ & \textstyle\sum_j A_{ij} x_j \geq b_i \\ & x_j \geq 0 \end{aligned}$$

Restricted Master Problem (RMP)

Prices (y)

$$\begin{aligned} \min\ & 1 - \textstyle\sum_i y_j z_i \\ \text{s.t.}\ & \textstyle\sum_i W_i z_i \leq L \\ & z_i \text{ integer} \end{aligned}$$

Pricing Problem (PP)

Patterns (z)

New column in A

Repeat while reduced cost is negative

Some loose ends

- Starting set of patterns
  - E.g. all patterns that contain exactly 1 order width
  - See later for another example

- Integer solution
  - Recall, RMP is LP problem
  - Rounding
  - Re-solve final Master Problem as ILP, instead of LP
- What if integer infeasible, or rounding not close to optimal?
  - Combine Column Generation with Branch-and-Bound
  - Branch-and-Price
  - See references

Application of GC to Network Planning
Problem statement

Given

- Topology (sources, grid sites, OXCs)
- Demand (job arrivals at sources)
- Survivability requirements (e.g. link and/or node failures)

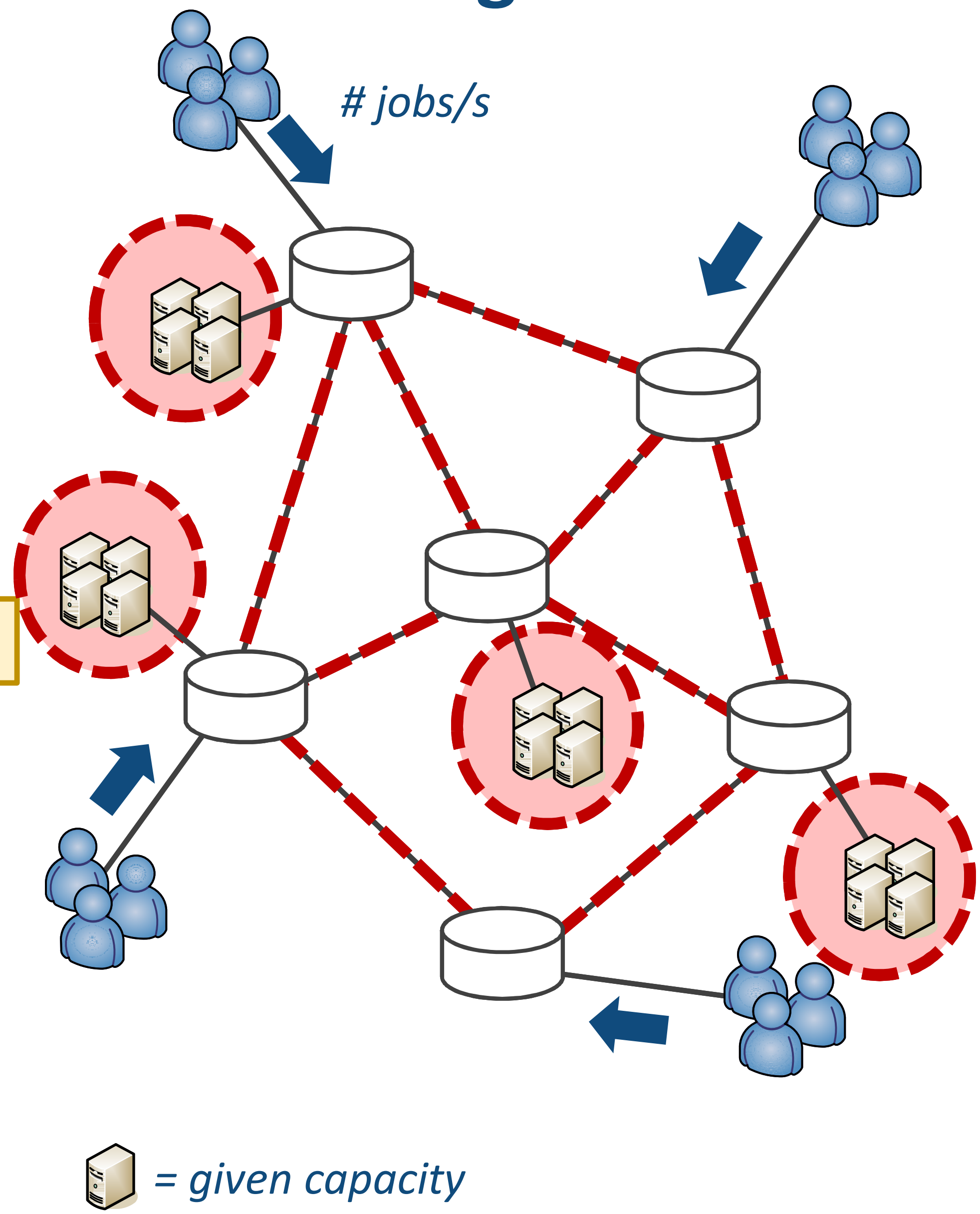
Shared path protection

Find

- Destinations sites and routes for each source
- Network and server capacity

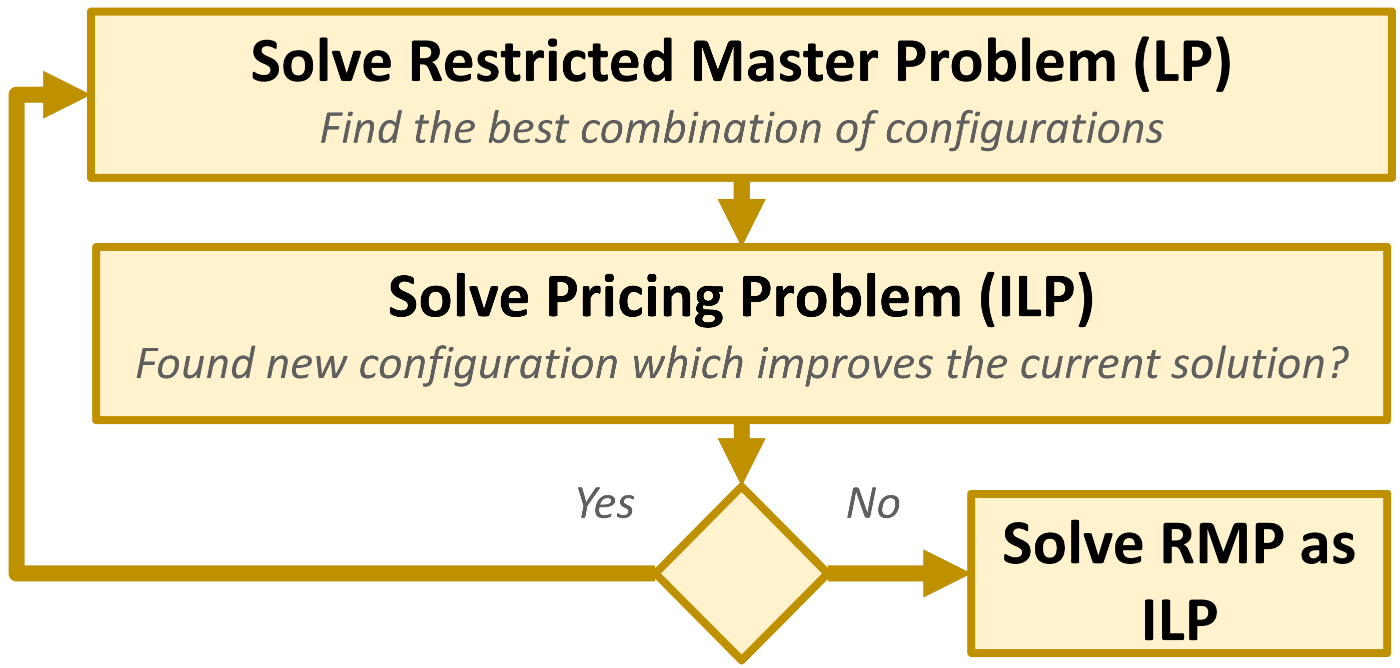
Such that

- Network and server resources are minimized



Application of GC to Network Planning

- Column generation:
  - Assume: given “configurations” = combination of working and backup paths
  - Restricted Master Problem (RMP) finds best combination of configurations
  - Pricing Problem (PP) finds new configuration that can reduce cost



Finding initial configurations for RMP

- Configuration associated with a particular <u>source</u>
  - Working path to primary destination
  - Backup path to secondary destination

- Algorithm:
  - Find shortest 2 disjoint paths from *s* to candidate destination(s): configuration *c*
  - Create new configurations that share backup links:
    - Remove working links of *c*
    - For all configurations *c'* with primary disjoint from *c*: set backup link weights to 0
    - Find backup path *b"* as shortest path to candidate destination in reduced graph
    - Create *c"*: primary of *c*, and *b"* as secondary

Suurballe's algorithm

Sharing of backup wavelengths

CPLEX Workflow

- "Your" way
  - Create LP model and write to .lp file in MATLAB or any other program that support it
  - Start session on CPLEX Interactive Optimizer
    - Import LP model
    - Set parameters
    - Solve
    - Retrieve variable values
- "My" way
  - CPLEX API supports: C, C++, Java, .NET and Python
  - Do everything in 1 programming language: pre-processing, create LP model, solver, post-processing
  - Unattended runs (e.g. parameter sweep)

Brief example (in Python)

```
# Make cplex package available
import cplex
# Create cplex object
cpx = cplex.Cplex()
# Parameter example - Limit tree size in memory to 8GB
# Rest of tree is saved on disk
cpx.parameters.mip.limits.treememory.set(8192)
cpx.parameters.mip.strategy.file.set(3)
# Objective function
cpx.objective.set_sense(self.cpx.objective.sense.minimize)
```

only specify direction, actual function is defined through declaring variables

Brief example (contd.)

lower and upper bounds (cplex.infinity is available)

```
# Create variables
cpx.variables.add(obj = [0], lb = [0.0],ub = [1.0],
                  types = [self.cpx.variables.type.binary])
```

contribution to objective function

continuous, integer, or binary

```
# Create constraints
# row_coeff contains weight of all variables in this constraint
cpx.linear_constraints.add(
        lin_expr = [len(row_coef), row_coef]],
        senses = ['G'], rhs = [1.0])
```

<, <=, =, >=, >

L, LE, E, GE, G

right-hand side of constraint

Brief example (contd.)

```
# Run solver
cpx.solve()

# Obtain objective value
solution = cpx.solution.get_objective_value()

# Obtain values of variables
variables = cpx.solution.pool.get_values(0)
```

array, i.e. access individual values through indexing

References

[1] R. L. Rardin, "Optimization in Operations Research," Prentice Hall, 1997.

[2] R. W. Haessler, "Selection and Design of Heuristic Procedures for Solving Roll Trim Problems," *Management Science*, vol. 34, no. 12, December 1988.

[3] M. E. Lübbecke, J. Desrosiers, "Selected Topics in Column Generation," *Operations Research*, vol. 53, no. 6, pp. 1007-1023, December 2005.

[4] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W.P. Savelsbergh, P. H. Vance, "Branch-and-Price: Column Generation for Solving Huge Integer Programs," *Operations Research*, vol. 46, no. 3, pp. 316-329, May-June 1998.

[5] IBM ILOG CPLEX Optimization Studio V12.2