COMMERCIAL MATHEMATICS

# LAZY CONSTRAINTS

## FOR FUN AND PROFIT

BAM 2016 – DR MICHAEL FORBES, ROBBIE PEARCE ET AL. THE UNIVERSITY OF QUEENSLAND

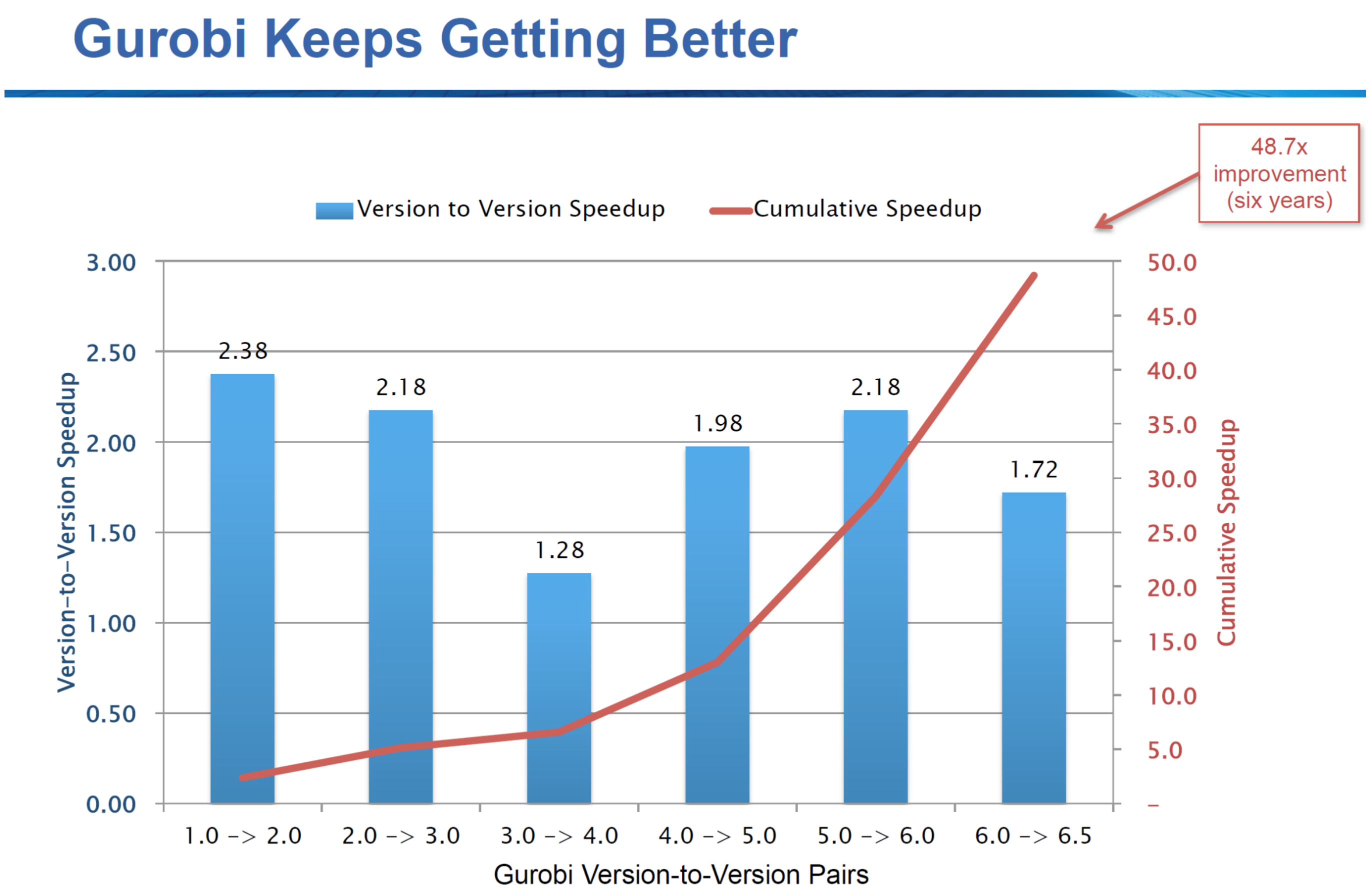
| AGENDA |
| --- |
| Solving tough MIP's |
| Branch and Price |
| Lazy Constraints |
| Fun |
| Profit |
| Benders Decomposition |
| Framework |

# Why we should persevere with a MIP approach to optimisation problems

- You get a bound on solution quality
- It makes you think about modelling the problem
- Someone else is dedicated to making your problem run faster
- Many ways to make MIPs faster
- Many MIP based heuristics.

1) BAM 2014

# Someone else is dedicated to making your problem run faster




1) Gurobi's Benchmark.pdf

# Many ways to make MIP faster

- Better modelling
- Lagrangian Relaxation
- Branch and Price
  - Branch, Price and Cut
- Lazy Constraints

# Branch and Price

- Reformulate the problem using some form of “composite” variable.
- At any LP relaxation, attempt to find variables which will improve the relaxation.
- Whenever we can’t find any, and the solution is not integer, branch (or cut).

1) Branch-and-Price: Column Generation for Solving Huge Integer Programs: Barnhart, Johnson, Nemhauser, Savelsbergh and Vance, 1996

# Branch and Price

## PROS

- Often a tighter LP relaxation
- Often reduced symmetry
- Separation into master and sub problems:
  - additional non-linear constraints in the sub-problem.
- Sometimes it's the only way to solve the problem.

## CONS

- Limited models where it is useful.
- Slow convergence of master problem.
- Tricks required for dual variable stabilisation.
- Not (yet) directly supported by modern IP solvers.

# Cuts

- MIP Solvers use Branch and Cut
  - Mysterious (proprietary) pre-processing phase first
- Solve LP relaxation
- If not integer feasible, try to add a cut:
  - An equality satisfied by any feasible integer solution but not satisfied by the current relaxed solution
- If it is too much work/not useful enough to find a cut, then branch
- “Users” (i.e. the modeller) can also add cuts on the fly

1) Outline of an algorithm for integer solutions to linear programs. Gomory (1958)

# Lazy Constraints

- Inequalities that cut off integer solutions
- Idea has existed for many years – e.g. Concorde TSP solver
- Often referred to as Branch and Cut (somewhat confusingly)
- Direct support by MIP packages has opened up many more options
  - Gurobi 5.0, May 2012
  - CPLEX?

1) Concorde Solver

# TSP with Lazy Constraints

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

$Subject\ to$:

$$\sum_{i} x_{ij} = 1\ \forall\ j$$

$$\sum_{i} x_{ji} = 1\ \forall\ j$$

$$\sum_{i \in S, j \in S} x_{ij} \leq |S| - 1\ \forall\ subsets\ S$$

# Lazy Constraints

**PROS**

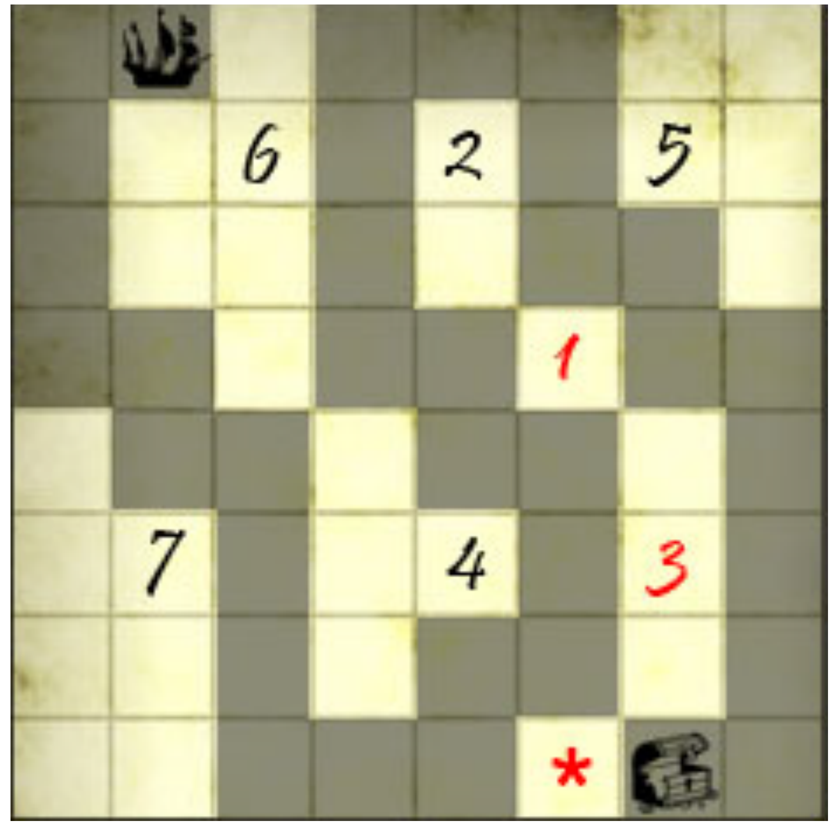
- Sometimes the only way to model the problem
- Much smaller models = much faster solve times
  - Optimistically assume constraints will be satisfied, add when not.
  - Optimistically estimate some component of the objective function and add constraints to improve our estimate.
- All the advantages of being embedded in a modern IP solver
  - multi-threading, heuristics, cuts, pre-solve, etc.

**CONS**

- Limited models where it is useful
- Can require too many lazy constraints

# Sometimes the only way to model the problem

- Pieces of 8
  - $x_{ijk} = 1$ if square $(i,j)$ is type $k$
  - Each square is used once
  - Squares of type 0 have exactly 2 neighbours of type 0 (except origin and destination have 1)
  - The right number of squares for each piece of 8
  - Pieces of different types aren't neighbours
  - At least one neighbour of the same type
- Plus …
  - Each piece of 8 is connected
  - There are no loops in the path.

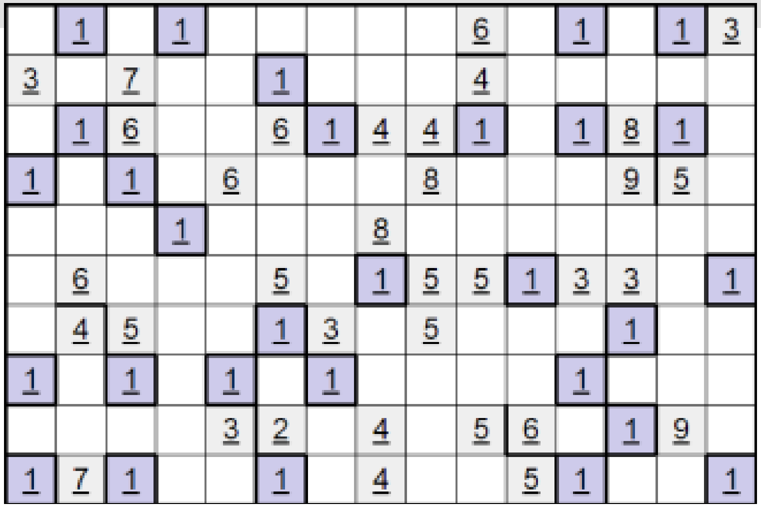


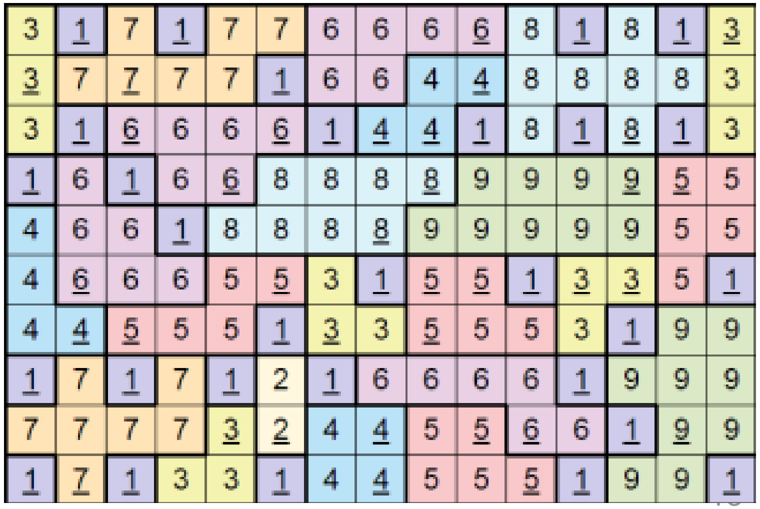

1) MUMS Puzzle Hunt 2011

# Sometimes the only way to model the problem

- Fillomino
  - $x_{ijk} = 1$ if square $(i,j)$ is type $k$
  - Each square is used once
- Plus …
  - Each "n-omino" has the correct number of pieces

- Can do this one with composite variables
  No 8$^{th}$ piece of unknown size





1) Fillomino, Pearce and Forbes, Submitted April 2016

# Optimistically assume constraints will be satisfied

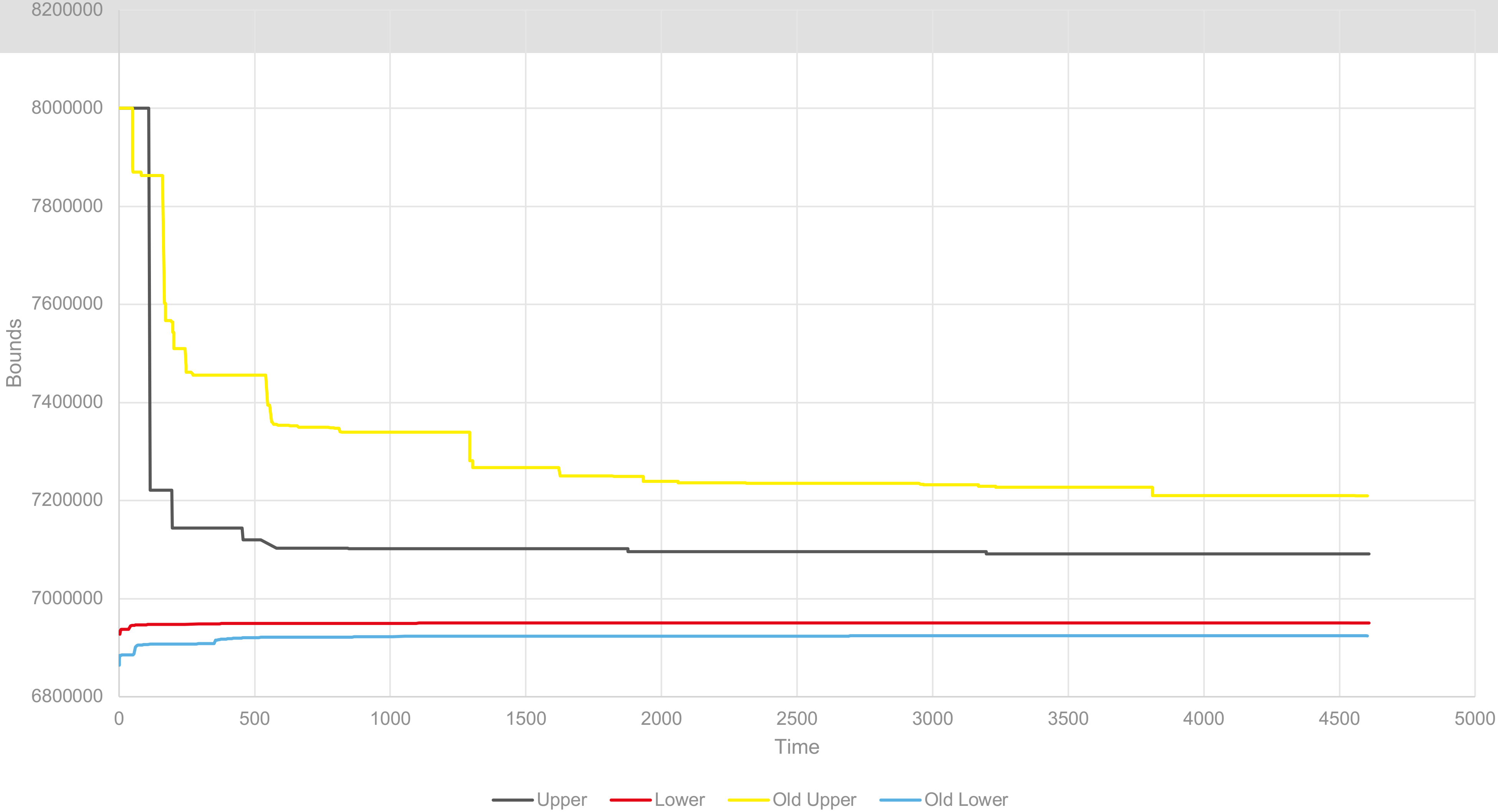

1) Optimizing Network Designs for the World's Largest Broadband Project. Ferris, Forbes, Forbes, Forbes and Kennedy, Interfaces, 2015

# Capacitated Spanning Tree Partition

| Direct IP Formulation | |
|---|---|
| $z_i$ | 1 if node *i* is a hub |
| $y_a$ | 1 if directional arc *a* is used in the solution |
| $x_a$ | "Units of demand" flowing on *a* |
| | Minimise cost of hubs and arc costs |

| Lazy IP Formulation | |
|---|---|
| $w_i$ | 1 is non demand node *i* is used in the solution |
| $y_a$ | 1 if non-directional *a* is used in the solution |
| | Number of hubs is (number of nodes used – number of arcs used) |
| | Lazily eliminate problems |

# New vs Old



8200000
8000000
7800000
7600000
7400000
7200000
7000000
6800000
Bounds
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
Time
Upper
Lower
Old Upper
Old Lower

# Pickup and Delivery Vehicle Routing

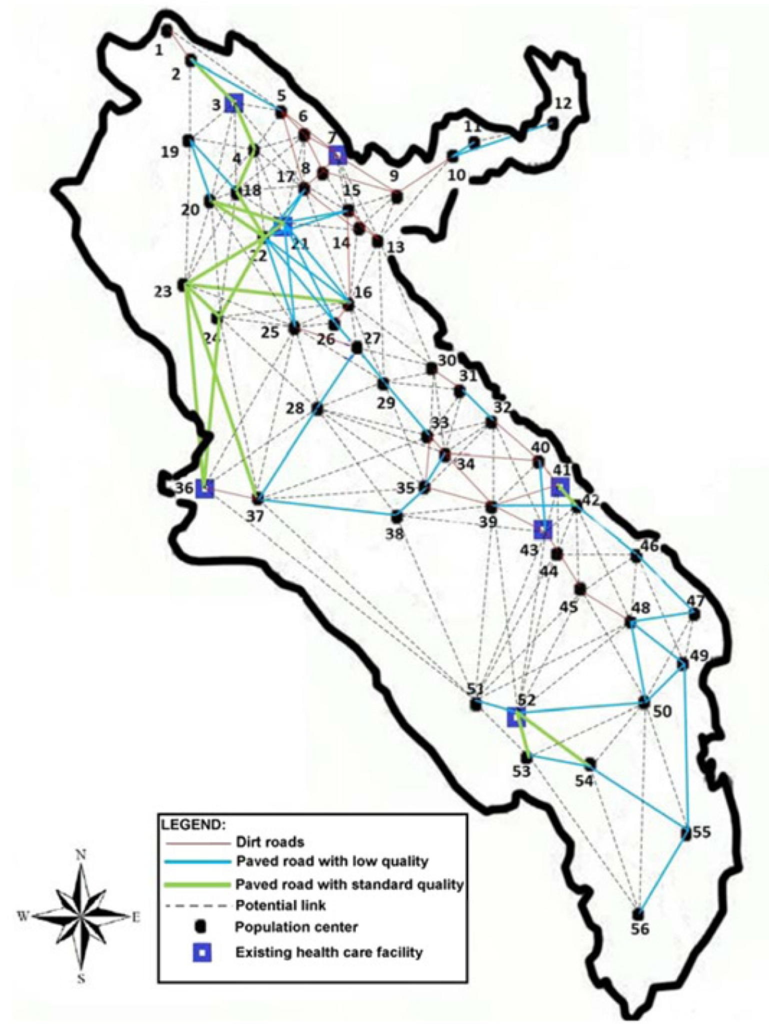
- Vehicle capacity and travel times
- Orders
  - Pickup and delivery locations
  - Pickup and delivery time window
- New approach combines composite variables and lazy constraints
- Composite Variable – Order String
  - Series of pickups and deliveries so that the vehicle starts and ends empty
  - Variables handle precedence (pickup before delivery), capacity and time windows
- Master problem sequences strings
  - Each order covered
  - Flow conservation through strings
  - Pairwise string connections legal
  - Longer string connections may be illegal – even cycles: Lazy Constraints

# GUFLNDP

- **G**eneral **U**ncapacitated **F**acility **L**ocation and **N**etwork **D**esign **P**roblem
- Nodes, Arcs, Facilities (a subset of nodes) $N, A, F$
- Cost of opening each facility (0 means already open)
- Cost of opening each arc (0 means already open)
- Cost per unit of movements on arcs
- Set of requests, each of which has a known volume, origin, candidate facilities (a subset of $F$)
- Any other constraints on the network structure.

1) UFL – many references
2) Modeling the budget-constrained dynamic uncapacitated facility location–network design problem … Ghaderi et al, 2013
3) An improved Benders decomposition algorithm for the tree of hubs location problem, Martin de Sa et al, 2013
4) Benders Decomposition for the Design of a Hub and Shuttle Public Transit System, Maheo et al, 2015


Kermanshah
Ivan
Illam
Shirvan
Lorestan
Mehran
Dareshahr
Dehloran
Abdanan
Iraq
Khozestan
LEGEND:
Inter. Boundary
Province Boundary
City Boundary
Province Capital
Urban Center


LEGEND:
Dirt roads
Paved road with low quality
Paved road with standard quality
Potential link
Population center
Existing health care facility
N
W
E
S

# GUFLNDP and Benders Decomposition

- $z$ variables open and close nodes
- $y$ variables open and close arcs
- Usual Benders Cut: $\theta_r \geq \theta^* - \sum \gamma_i z_i - \sum \lambda_{ij} y_{ij}$.
  - Cost for request $r$ is greater equal to current cost (solving sub-problem) minus saving for opening candidate facilities minus saving for opening closed arcs
  - Can solve flow problem for each resource and use dual variables on node and arc constraints
- Solve sub-problem and use dual variables form appropriate constraints

1) Partitioning procedures for solving mixed-variables programming problems, Benders 1962

# Pareto-optimality

- An undominated cut is said to be Pareto-Optimal
- Magnanti and Wong core point method has been popular – but it is slow to compute this
- "Natural Benders Cut" for GUFLNDP
  - Solve shortest path tree from origin to nearest open facility
  - All dual variables for open facilities and arcs set to zero
  - Dual variables for node $i$ set to $\max(\theta^* - dist_i, d_i^*)$ where $dist_i$ is the shortest distance from the origin to node $i$ and $d_i^*$ is the shortest possible distance from node $i$ to any candidate facility for the current resource, if all arcs were opened.
  - Dual variables for unopened arcs set to satisfy the duality conditions:

$$\lambda_{ij} = \max(0, \gamma_i - \gamma_j - c_{ij})$$

1) Accelerating Benders Decomposition: Algorithmic Enhancement and Model Selection Criteria, Magnanti and Wong, 1981

# Pareto-optimality

- The Natural Cut:
  - Is dual feasible
  - Has the same objective value as the primal optimal
  - Therefore it is primal optimal and is a valid Benders Cut
- Proving Pareto-Optimality of the Natural Cut
  - Carefully select solutions where the cut is binding (equals optimal solution of sub-problem)
  - Show that any cut that has equality at all these solutions is the natural cut
  - Therefor the cut can't be dominated
  - The only exception comes from connecting a resource to its closest candidate facility
  - Can solve this with a Balinski style cut (second closest candidate)


1. Integer programming: Methods, uses, computation. Balinski, 1965
2. Pareto-Optimality of the Balinski Cut for the Uncapacitated Facility Location Problem, Watson and Rogers, 2007

# Benders Summary

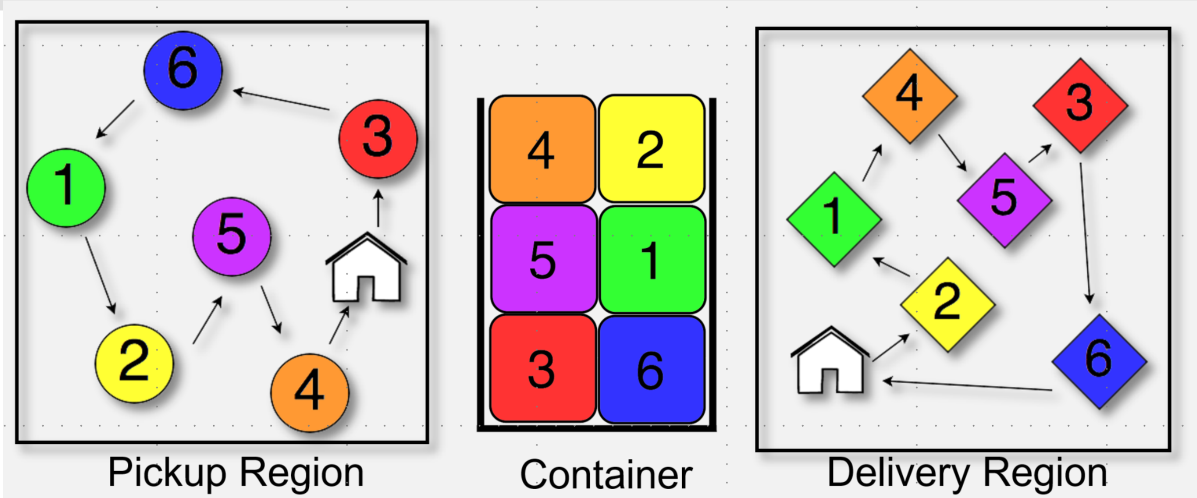
- Works well for problems which disaggregate (no shared capacity)
- Avoid feasibility cuts
- Warm start usually helps
- Pareto Optimal cuts important
  - Often naturally achievable
- Heuristics important
- Open issues
  - Natural warm start
  - Warm start in callback
  - Level of disaggregation
  - Natural second best cut is also pareto-optimal – which is best?

1) Disaggregated Benders Decomposition for solving a Network Maintenance Scheduling Problem, Pearce and Forbes, 2016
2) Several more papers to appear. Pearce at al, 2016


Why don't we have both?

# Other Applications

- Pickup and delivery TSP Multiple Stacks
  - Solve two TSPs
  - Add lazy constraints if not stack feasible
  - Tightness of constraints!




- Power flow models
  - Many quadratic components in objective (per arc, wire type, time period)
  - Approximate with “natural” cut: $\theta_{ijwt} \geq {x^*_{ijwt}}^2 + 2x^*_{ijwt}(x_{ijwt} - x^*_{ijwt})$
- Crane movements
  - Lower bound IP ignoring clash constraints
  - In call back, resolve clash constraints (another IP) to get upper bound and cut off all similar solutions

# Conclusions

- Solve the model you want to solve
  - Especially if there is a big gap from “structure” to full detail
- Can you approximate the objective function and leave out detail?
- Is there a natural way to refine an approximation / improve feasibility?
- Tightness of cuts is important
  - Lift LHS, tighten RHS
- Become an expert in your modelling environment
  - Fast prototyping, especially of Callbacks

- Questions?