Constraint Programming and Mathematical Programming

Tutorial

John Hooker

Carnegie Mellon University

CP02

September 2002

This tutorial is available at

http://ba.gsia.cmu.edu/jnh

g

Why Integrate CP and MP?
Search/Inference Duality
Decomposition
Relaxation
Putting It Together
Other OR Methods of Interest to CP
Surveys/Tutorials on Hybrid Methods

Programming ≠ Programming

- Constraint *programming* is related to computer programming.

- Mathematical *programming* has nothing to do with computer programming.
  - "Programming" historically refers to logistics plans (George Dantzig's first application).
  - MP is purely declarative.

Why Integrate CP and MP?

Eventual goal:
View CP and MP as special cases of a general method

Motivation to Integrate CP and MP

- Inference + relaxation.
  - CP's inference techniques tend to be effective when constraints contain few variables.
    - Misleading to say CP is effective on "highly constrained" problems.
  - MP's relaxation techniques tend to be effective when constraints or objective function contain many variables.
    - For example, cost and profit.

Motivation to Integrate CP and MP

- "Horizontal" + "vertical" structure.
  - CP's idea of global constraint exploits structure within a problem (horizontal structure).
  - MP's focus on special classes of problems is useful for solving relaxations or subproblems (vertical structure).

Motivation to Integrate CP and MP

- Procedural + declarative.
  - Parts of the problem are best expressed in MP's declarative (solver-independent) manner.
  - Other parts benefit from search directions provided by user.

Integration Schemes

Recent work can be broadly seen as using four integrative ideas:

- *Double modeling* - Use both CP and MP models and exchange information while solving.
- *Search-inference duality* - View CP and MP methods as special cases of a search/inference duality.
- *Decomposition* - Decompose problems into a CP part and an MP part using a Benders scheme.
- *Relaxation* - Exploit the relaxation technology of OR (applies to all of the above).

Double Modeling

- Write part of the problem in CP, part in MP, part in both.
- Exchange information - bounds, infeasibility, etc.
- *Dual modeling is a feature of other more specific schemes and will not be considered separately.*

Search-Inference Duality

- CP and MP have a fundamental isomorphism: search and inference work together in a duality relationship, such as branch and infer *(Bockmayr & Kasper)*.
  - *Search* (a primal method) examines possible solutions.
    - Branching (domain splitting), local search, etc.
  - *Inference* (a dual method) deduces facts from the constraint set.
    - Domain reduction (CP), cutting planes (MP).
  - Both the search and inference phases can combine CP/MP.

Decomposition

- Some problems can be decomposed into a master problem and subproblem.
  - Master problem searches over some of the variables.
  - For each setting of these variables, subproblem solves the problem over the remaining variables.
  - One scheme is a generalized Benders decomposition.
    - CP is natural for subproblem, which can be seen as an inference (dual) problem.

Relaxation

- MP relies heavily on relaxations to obtain bounds on the optimal value.
  - Continous relaxations, Lagrangean relaxations, etc.
- Global constraints can be associated with relaxations as well as filtering algorithms.
  - This can prune the search tree in a branch-and-infer approach.
- Relaxation of subproblem can dramatically improve decomposition if included in master problem.

Search/Inference Duality

Linear Programming
Integer Programming
Gomory Cuts (dual method)
Branch and Infer
Discrete Lot Sizing

Example: Linear Programming Duality

(Dantzig, Von Neumann)

Primal problem:

$$\begin{aligned} \text{minimize} \quad & cx \\ \text{subject to} \quad & Ax \geq b \quad (u) \\ & x \geq 0 \end{aligned}$$

Dual problem:

$$\begin{aligned} \text{maximize} \quad & ub \\ \text{subject to} \quad & uA \leq c \quad (x) \\ & u \geq 0 \end{aligned}$$

Primal: *Search* for optimal value x^* of x.

Dual: *Prove* that cx^* is optimal value of primal:

Identify a linear combination u of the primal constraints

$$uAx \geq ub$$

$\wedge$| |$\vee$

that implies $cx \geq cx^*$

Example: Linear Programming Duality

- In this case, the solution of the dual has polynomial size.
  - LP belongs to NP and co-NP.
  - Dual is itself an LP.
  - LP can be solved by primal-dual algorithm.
- In general, the dual solution tends to be exponential in size.
  - For example, integer programming.

Example: Integer Programming

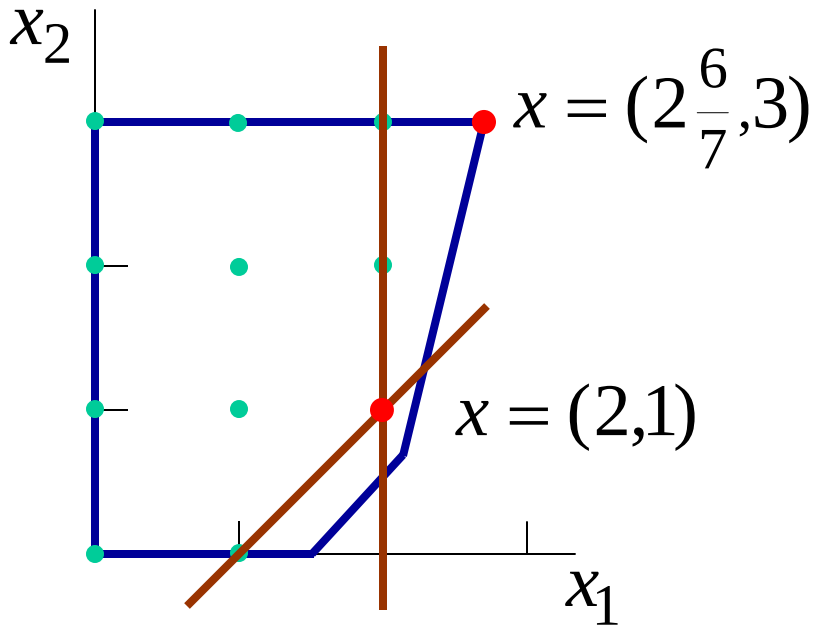
Example from Wolsey, 1998.

$$
\begin{aligned}
\text{maximize} \quad & 4x_1 - x_2 \\
\text{subject to} \quad & 7x_1 - 2x_2 \le 14 \\
& x_2 \le 3 \\
& 2x_1 - 2x_2 \le 3 \\
& x_1, x_2 \ge 0 \text{ and integer}
\end{aligned}
$$

- Can be solved by branching on values of x_j (to be illustrated shortly). This is a primal method.
- Can also be solved by generating cutting planes. This is a dual method.

A Cutting Plane Approach
(Gomory)

- Solve continuous relaxation of problem.
- Generate Gomory cuts.
- Re-solve relaxation (solution is now integer).



• The problem with the two cutting planes is:

$$
\begin{aligned}
\text{maximize} \quad & 4x_1 - x_2 \\
\text{subject to} \quad & 7x_1 - 2x_2 \le 14 \\
& x_2 \le 3 \\
& 2x_1 - 2x_2 \le 3 \\
& x_1 - x_2 \le 1 \\
& x_1 \le 2 \\
& x_1, x_2 \ge 0
\end{aligned}
$$

Gomory cuts →

Solution of this relaxation is $(x_1,x_2) = (2,1)$

• Can now solve the original problem by solving continuous relaxation (a linear programming problem).

• In the worst case, cutting plane proofs are exponentially long.

• In practice, cutting planes are combined with branching *(branch and cut).*

These Gomory cuts are *rank 1 Chvátal* cuts obtained by taking a nonnegative linear combination of constraints and rounding.

maximize	$4x_1 - x_2$	
subject to	$7x_1 - 2x_2 \le 14$	1
	$x_2 \le 3$	2
	$2x_1 - 2x_2 \le 3$	0
	$x_1 - x_2 \le 1$	0
	$x_1 \le 2$	0
	$x_1, x_2 \ge 0$	0

Multipliers in linear combination

$$7x_1 \le 20$$

Can round down RHS → $x_1 \le 2\frac{6}{7}$

Cutting plane → $x_1 \le 2$

Do this recursively to generate all valid cutting planes & convex hull. *(Chvátal)*

Branch and Infer

$$
\begin{array}{ll}
\min & 5x_1 + 8x_2 + 5x_3 \\
\text{subject to} & 3x_1 + 5x_2 + 3x_3 \geq 30 \\
& \text{all-different}\{x_1, x_2, x_3\} \\
& x_j \in \{1, \ldots, 4\}
\end{array}
$$

We will illustrate how search and inference may be combined to solve this problem by:

- constraint programming
- integer programming
- a hybrid approach

Solve as a constraint programming problem

Search: Domain splitting

Inference: Domain reduction and constraint propagation

$$5x_1 + 8x_2 + 4x_3 \leq z$$
$$3x_1 + 5x_2 + 2x_3 \geq 30$$
$$\text{all-different}\{x_1, x_2, x_3\}$$
$$x_j \in \{1, \ldots, 4\}$$

Start with z = ∞.
Will decrease as feasible solutions are found.

Domain reduction for inequalities

- Bounds propagation on $5x_1 + 8x_2 + 4x_3 \leq z$
  $3x_1 + 5x_2 + 2x_3 \geq 30$

For example, $3x_1 + 5x_2 + 2x_3 \geq 30$ implies

$$x_2 \geq \frac{30 - 3x_1 - 2x_3}{5} \geq \frac{30 - 12 - 8}{5} = 2$$

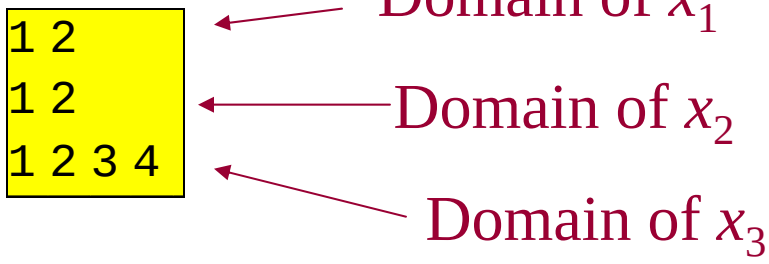
So the domain of x_2 is reduced to $\{2,3,4\}$.

Domain reduction for all-different *(e.g., Régin)*

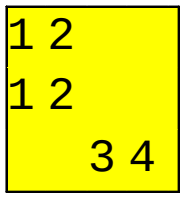
- Maintain hyperarc consistency on

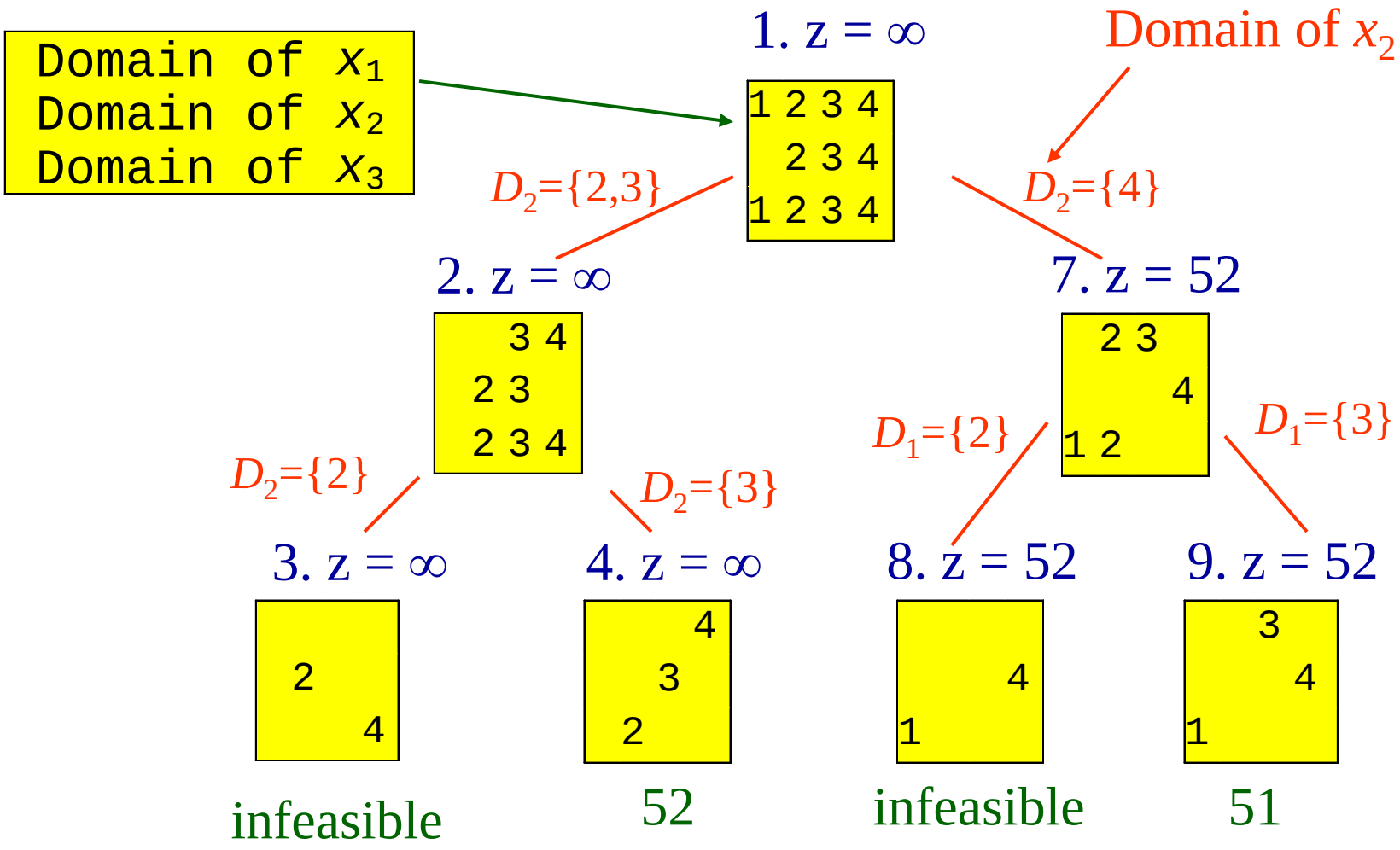
$$\text{all-different}\{x_1, x_2, x_3\}$$

Suppose for example:



Then one can reduce the domains:




Domain of x_1
Domain of x_2
Domain of x_3
1. z = ∞
1 2 3 4
2 3 4
1 2 3 4
Domain of x_2
D_2={2,3}
D_2={4}
2. z = ∞
3 4
2 3
2 3 4
7. z = 52
2 3
4
1 2
D_2={2}
D_2={3}
D_1={2}
D_1={3}
3. z = ∞
2
4
4. z = ∞
4
3
2
8. z = 52
4
1
9. z = 52
3
4
1
infeasible
52
infeasible
51

Solve as an integer programming problem

Search: Branch on variables with fractional values in solution of continuous relaxation.
Inference: Generate cutting planes (not illustrated here).

Rewrite problem using integer programming model:

Let y_{ij} be 1 if $x_i = j$, 0 otherwise.

$$
\begin{aligned}
\text{min} \quad & 4x_1 + 3x_2 + 5x_3 \\
\text{subject to} \quad & 4x_1 + 2x_2 + 4x_3 \geq 17 \\
& x_i = \sum_{j=1}^{5} j y_{ij}, \quad i = 1,2,3 \\
& \sum_{j=1}^{4} y_{ij} = 1, \quad i = 1,2,3 \\
& y_{ij} \in \{0,1\}, \quad \text{all } i, j
\end{aligned}
$$

Continuous relaxation

Use a linear programming algorithm to solve a continuous relaxation of the problem at each node of the search tree to obtain a lower bound on the optimal value of the problem at that node.

$$\begin{aligned}
\text{min} \quad & 4x_1 + 3x_2 + 5x_3 \\
\text{subject to} \quad & 4x_1 + 2x_2 + 4x_3 \geq 17 \\
& x_i = \sum_{j=1}^{4} j y_{ij}, \quad i = 1,2,3 \\
& \sum_{j=1}^{4} y_{ij} = 1, \quad i = 1,2,3 \\
& 0 \leq y_{ij} \leq 1, \quad \text{all } i, j
\end{aligned}$$

Relax integrality

Branch and bound (Branch and relax)

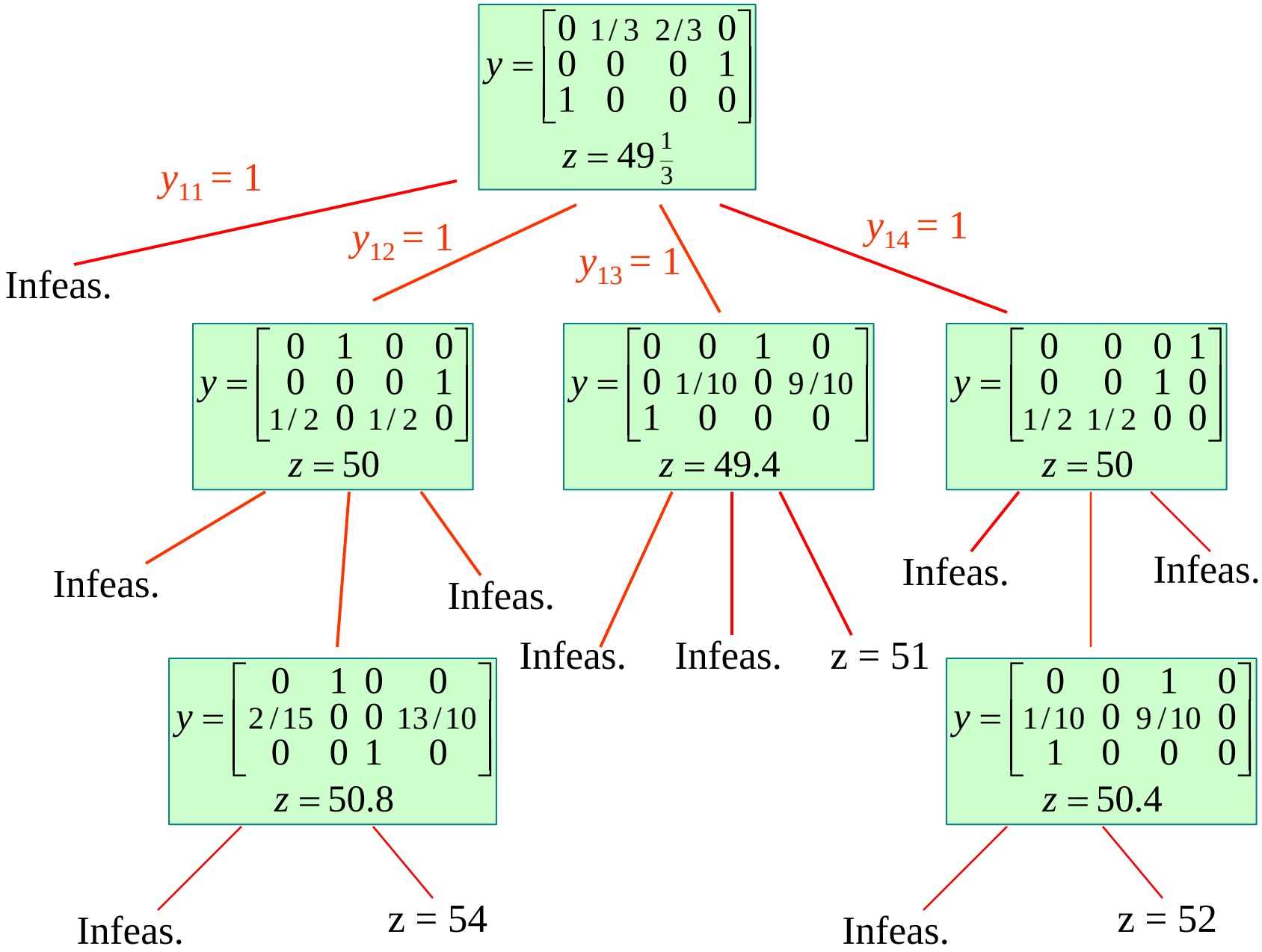
The *incumbent solution* is the best feasible solution found so far.

At each node of the branching tree:

- If Optimal value of relaxation $\geq$ Value of incumbent solution

There is no need to branch further.

- No feasible solution in that subtree can be better than the incumbent solution.


$y = \begin{bmatrix} 0 & 1/3 & 2/3 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$
$z = 49\frac{1}{3}$
$y_{11} = 1$
$y_{12} = 1$
$y_{13} = 1$
$y_{14} = 1$
Infeas.
$y = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1/2 & 0 & 1/2 & 0 \end{bmatrix}$
$z = 50$
$y = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1/10 & 0 & 9/10 \\ 1 & 0 & 0 & 0 \end{bmatrix}$
$z = 49.4$
$y = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1/2 & 1/2 & 0 & 0 \end{bmatrix}$
$z = 50$
Infeas.
Infeas.
Infeas.
Infeas.
Infeas.
Infeas.
z = 51
$y = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 2/15 & 0 & 0 & 13/10 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
$z = 50.8$
$y = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1/10 & 0 & 9/10 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$
$z = 50.4$
Infeas.
z = 54
Infeas.
z = 52

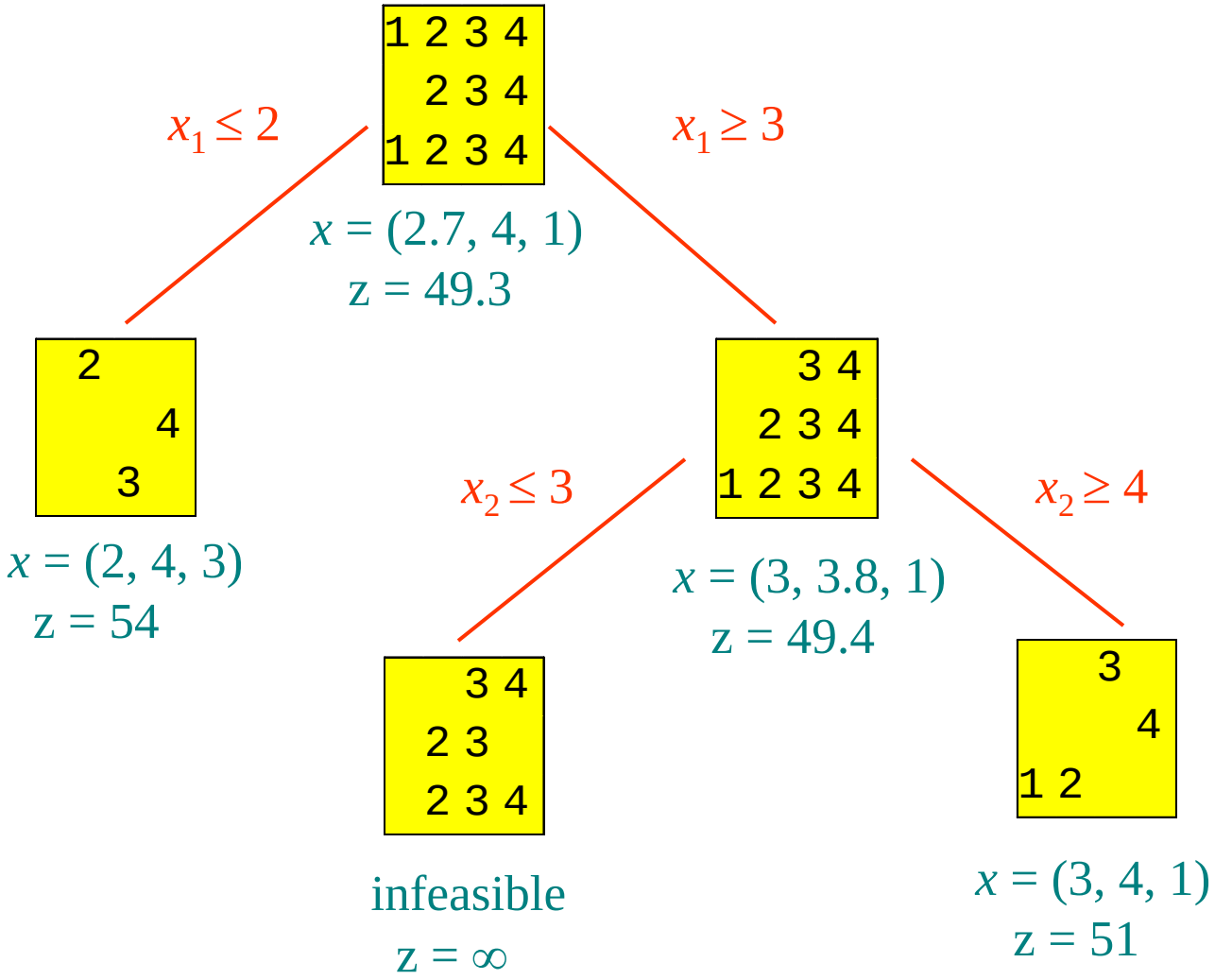
Solve using a hybrid approach

Search:

- Branch on fractional variables in solution of knapsack constraint relaxation $3x_1 + 5x_2 + 2x_3 \geq 30$.
  - Do not relax constraint with y_{ij}'s. This makes relaxation too large without much improvement in quality.
  - If variables are all integral, branch by splitting domain.
- Use branch and bound.

Inference:

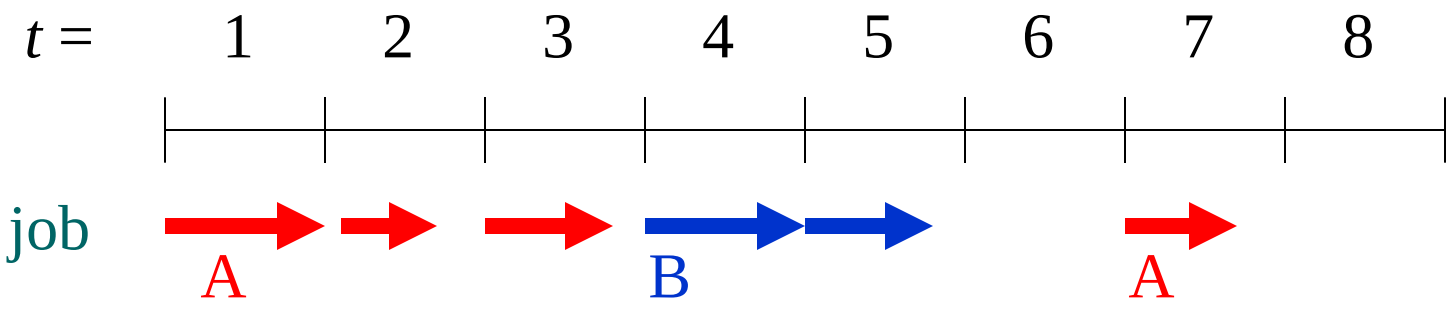
- Use bounds propagation.
- Maintain hyperarc consistency for all-different.


1 2 3 4
2 3 4
1 2 3 4
$x_1 \le 2$
$x_1 \ge 3$
x = (2.7, 4, 1)
z = 49.3
2
4
3
x = (2, 4, 3)
z = 54
3 4
2 3 4
1 2 3 4
$x_2 \le 3$
$x_2 \ge 4$
x = (3, 3.8, 1)
z = 49.4
3 4
2 3
2 3 4
infeasible
z = ∞
3
4
1 2
x = (3, 4, 1)
z = 51

Discrete Lot Sizing

- Manufacture at most one product each day.
- When manufacturing starts, it may continue several days.
- Switching to another product incurs a cost.
- There is a certain demand for each product on each day.
- Products are stockpiled to meet demand between manufacturing runs.
- Minimize inventory cost + changeover cost.

Discrete lot sizing example



y_t = A A A B B 0 A 0

0 = dummy job

IP model

(Wolsey)

$$\min \quad \sum_{t,i}\left(h_{it}s_{it} + \sum_{j\neq i} q_{ij}\delta_{ijt} \right)$$

s.t.

$$s_{i,t-1} + x_{it} = d_{it} + s_{it}, \quad \text{all } i,t$$

$$z_{it} \geq y_{it} - y_{i,t-1}, \quad \text{all } i,t$$

$$z_{it} \leq y_{it}, \quad \text{all } i,t$$

$$z_{it} \leq 1 - y_{i,t-1}, \quad \text{all } i,t$$

$$\delta_{ijt} \geq y_{i,t-1} + y_{jt} - 1, \quad \text{all } i,t$$

$$\delta_{ijt} \geq y_{i,t-1}, \quad \text{all } i,t$$

$$\delta_{ijt} \leq y_{jt}, \quad \text{all } i,t$$

$$x_{it} \leq C y_{it}, \quad \text{all } i,t$$

$$\sum_{i} y_{it} = 1, \quad \text{all } t$$

$$y_{it}, z_{it}, \delta_{ijt} \in \{0,1\}$$

$$x_{it}, s_{it} \geq 0$$

Hybrid model

total inventory + changeover cost

$$\min \quad \sum_t (u_t + v_t)$$

stock level

$$\text{s.t.} \quad u_t \geq \sum_i h_i s_{it}, \ \text{all } t$$

changeover cost

$$v_t \geq q_{y_{t-1} y_t}, \ \text{all } t$$

daily production

$$s_{i,t-1} + x_{it} = d_{it} + s_{it}, \ \text{all } i,t$$

inventory balance

$$0 \leq x_{it} \leq C, \ s_{it} \geq 0, \ \text{all } i,t$$

$$(y_t \neq i) \rightarrow (x_{it} = 0), \ \text{all } i,t$$

To create relaxation:

Put into relaxation

Generate inequalities to put into relaxation (to be discussed)

$$\min \quad \sum_t (u_t + v_t)$$

$$\text{s.t.} \quad u_t \geq \sum_i h_i s_{it}, \ \text{all } t$$

$$v_t \geq q_{y_{t-1} y_t}, \ \text{all } t$$

$$s_{i,t-1} + x_{it} = d_{it} + s_{it}, \ \text{all } i,t$$

$$0 \leq x_{it} \leq C, \ s_{it} \geq 0, \ \text{all } i,t$$

$$(y_t \neq i) \rightarrow (x_{it} = 0), \ \text{all } i,t$$

Apply constraint propagation to everything

Solution

Search: Domain splitting, branch and bound using relaxation of selected constraints.

Inference: Domain reduction and constraint propagation.

Characteristics:

- Conditional constraints impose consequent when antecedent becomes true in the course of branching.
- Relaxation is somewhat weaker than in IP because logical constraints are not all relaxed.
- But LP relaxations are much smaller--quadratic rather than cubic size.
- Domain reduction helps prune tree.

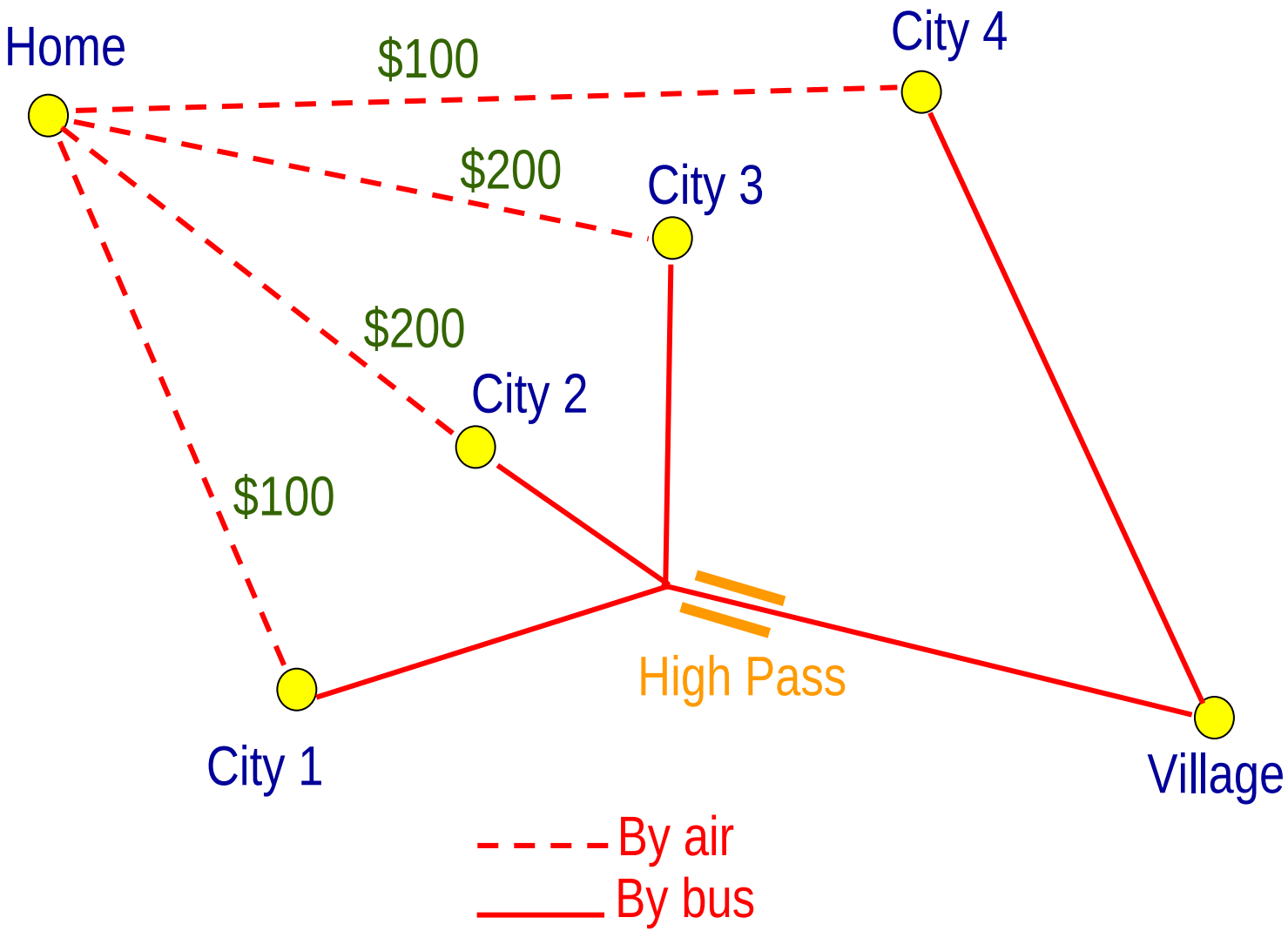
Decomposition

Idea of Benders Decomposition
Benders in the Abstract
Classical Benders
Logic Circuit Verification
Machine Scheduling

Idea Behind Benders Decomposition
"Learn from one's mistakes."

- Distinguish primary variables from secondary variables.
- Search over primary variables (*master problem*).
- For each trial value of primary variables, solve problem over secondary variables (*subproblem*).
- Can be viewed as solving a subproblem to generate nogoods (*Benders cuts*) in the primary variables.
- Add the Benders cut to the master problem and re-solve.
- Can also be viewed as projecting problem onto primary variables.

A Simple Example
Find Cheapest Route to a Remote Village

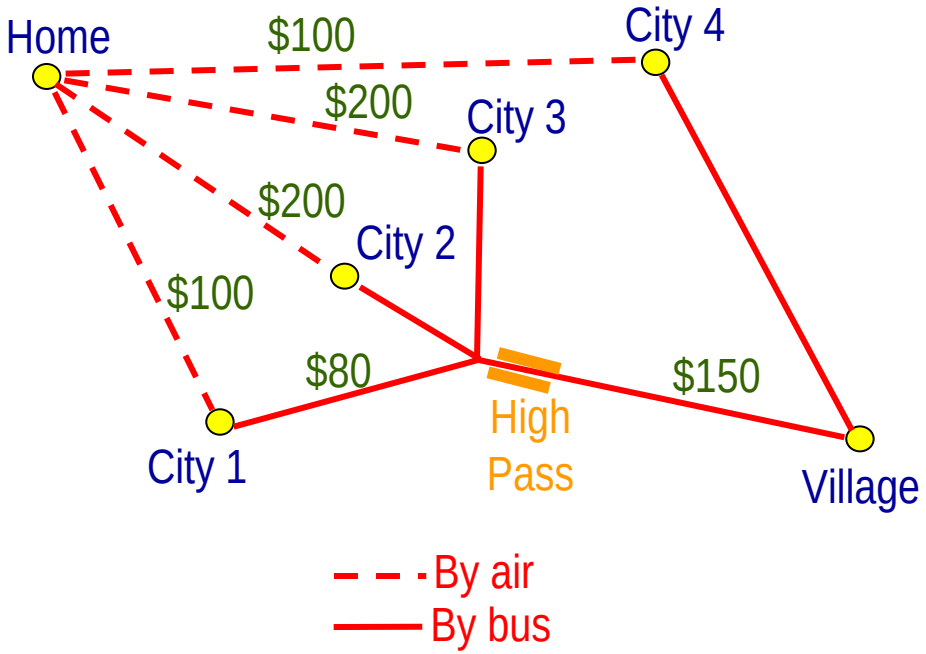


Let x = flight destination
y = bus route

Find cheapest route (x,y)

Begin with x = City 1 and pose the subproblem:

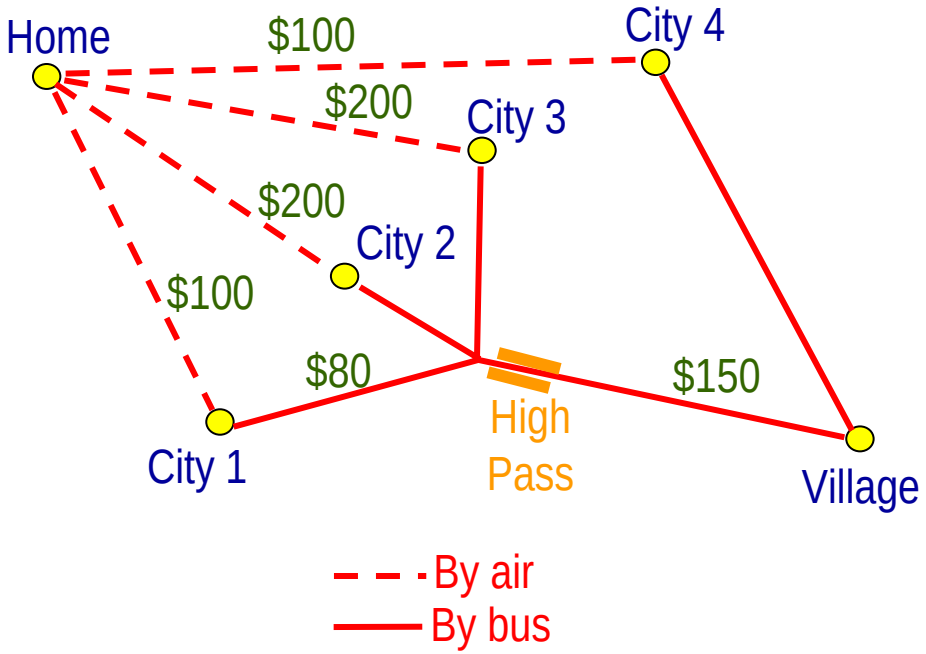
Find the cheapest route given that x = City 1.
Optimal cost is \$100 + 80 + 150 = \$330.



The "dual" problem of finding the optimal route is to prove optimality. The "proof" is that the route from City 1 to the village must go through High Pass. So

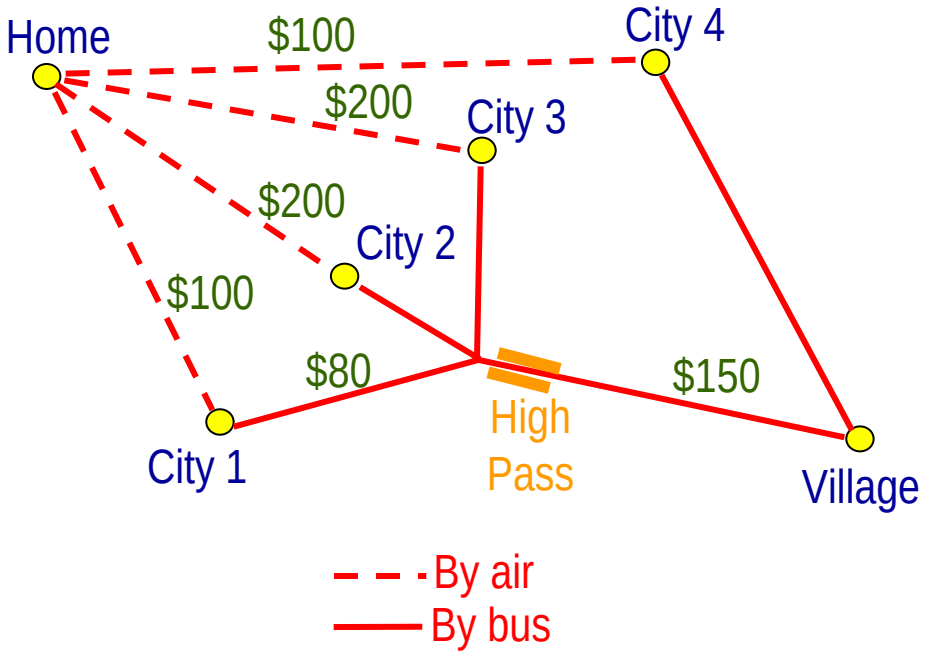
cost $\geq$ airfare + bus from city to High Pass + $150

But *this same argument* applies to City 1, 2 or 3. This gives us the above Benders cut.



Specifically the Benders cut is

$$\text{cost} \geq B_{\text{City}\,1}(x) = \begin{cases} \$100 + 80 + 150 & \text{if } x = \text{City}\,1 \\ \$200 + 150 & \text{if } x = \text{City}\,2,3 \\ \$100 & \text{if } x = \text{City}\,4 \end{cases}$$



Now solve the *master problem:*

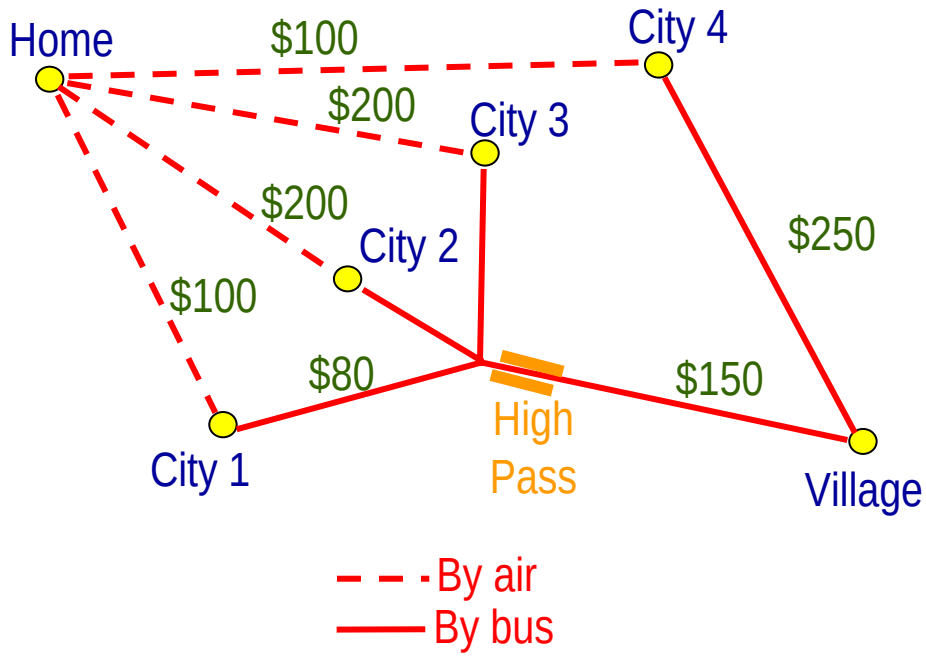
Pick the city x to minimize cost subject to

$$\text{cost} \geq B_{\text{City}\,1}(x) = \left\{ \begin{array}{ll} \$100 + 80 + 150 & \text{if } x = \text{City}\,1 \\ \$200 + 150 & \text{if } x = \text{City}\ 2,3 \\ \$100 & \text{if } x = \text{City}\ 4 \end{array} \right\}$$

Clearly the solution is x = City 4, with cost $100.

Now let x = City 4 and pose the subproblem:

Find the cheapest route given that x = City 4.
Optimal cost is \$100 + 250 = \$350.



Again solve the master problem:

Pick the city x to minimize cost subject to

$$\text{cost} \geq B_{\text{City }1}(x) = \begin{cases} \$100 + 80 + 150 & \text{if } x = \text{City } 1 \\ \$200 + 150 & \text{if } x = \text{City } 2,3 \\ \$100 & \text{if } x = \text{City } 4 \end{cases}$$

$$\text{cost} \geq B_{\text{City }4}(x) = \begin{cases} \$350 & \text{if } x = \text{City } 1 \\ \$0 & \text{otherwise} \end{cases}$$

The solution is x = City 1, with cost \$330. Because we found a feasible route with this cost, we are done.

Benders Decomposition in the Abstract

$$\min_{x,y} \quad f(x,y)$$
$$\text{s.t.} \quad C(x,y)$$

Secondary variables

Primary variables

For a given value $\bar{x}$ of x, solve the subproblem:

$$\min_{y} \quad f(\bar{x},y)$$
$$\text{s.t.} \quad C(\bar{x},y)$$

Let y^* be an optimal solution with optimal value v^*. To find a Benders cut, consider the *inference dual:*

$$\max_{y,v} \quad v$$
$$\text{s.t.} \quad C(\bar{x},y) \rightarrow f(\bar{x},y) \geq v$$

The inference dual clearly has the same optimal value v^*.

The solution of the inference dual is a *proof* that $f(\bar{x}, y) \geq v^*$ follows from $C(\bar{x}, y)$

Thus when $x = \bar{x}$ we have a proof that $f(x, y)$ is at least v^*

We want to use this *same proof schema* to derive that $f(x, y)$ is at least $B_{y^*}(x)$ for any x. (In particular $B_{y^*}(\bar{x}) = v^*$.)

To find a better solution than v^* we solve the *master problem*

$$\min_{x,v} \quad v$$

$$\text{s.t.} \quad v \geq B_{x^*}(x)$$

Benders cut

At iteration K+1 the master problem is

$$\begin{aligned} \min_{x,v} \quad & v \\ \text{s.t.} \quad & v \geq B_{x^k}(x), k = 0,\ldots,K \end{aligned}$$

Where $x^1,\ldots,x^K$ are the solutions of the first K master problems.

Continue until the subproblem has the same optimal value as the previous master problem.

Classical Benders Decomposition

(Benders)

$$\begin{aligned} \min_{x,y} \quad & f(x) + cy \\ \text{s.t.} \quad & g(x) + Ay \geq a \\ & y \geq 0 \\ & x \in D,\ y \in R^n \end{aligned}$$

For a given $\bar{x}$ the subproblem is the LP

$$\begin{aligned} \min_{y} \quad & f(\bar{x}) + cy \\ \text{s.t.} \quad & Ay \geq a - g(\bar{x}) \quad (u) \\ & y \geq 0 \end{aligned}$$

Dual variables

With optimal solution x^* and optimal value v^*.

The inference dual in this case is the classical LP dual

$$\begin{aligned} \min_{u} \quad & u(a - g(\bar{x})) \\ \text{s.t.} \quad & uA \le c \\ & u \ge 0 \end{aligned}$$

The dual solution u^* provides a proof that z^* is the optimal value: the linear combination $u^* Ay \ge u^*(a - g(\bar{x}))$ of the primal constraints dominates $cy \ge z^*$

Note that u^* is *dual feasible for any* x. So by weak duality, $u^*(a - g(x))$ is a lower bound on the optimal value of the subproblem for any x. So we have the Benders cut,

$$v \ge f(x) + u^*(a - g(x))$$

The master problem is

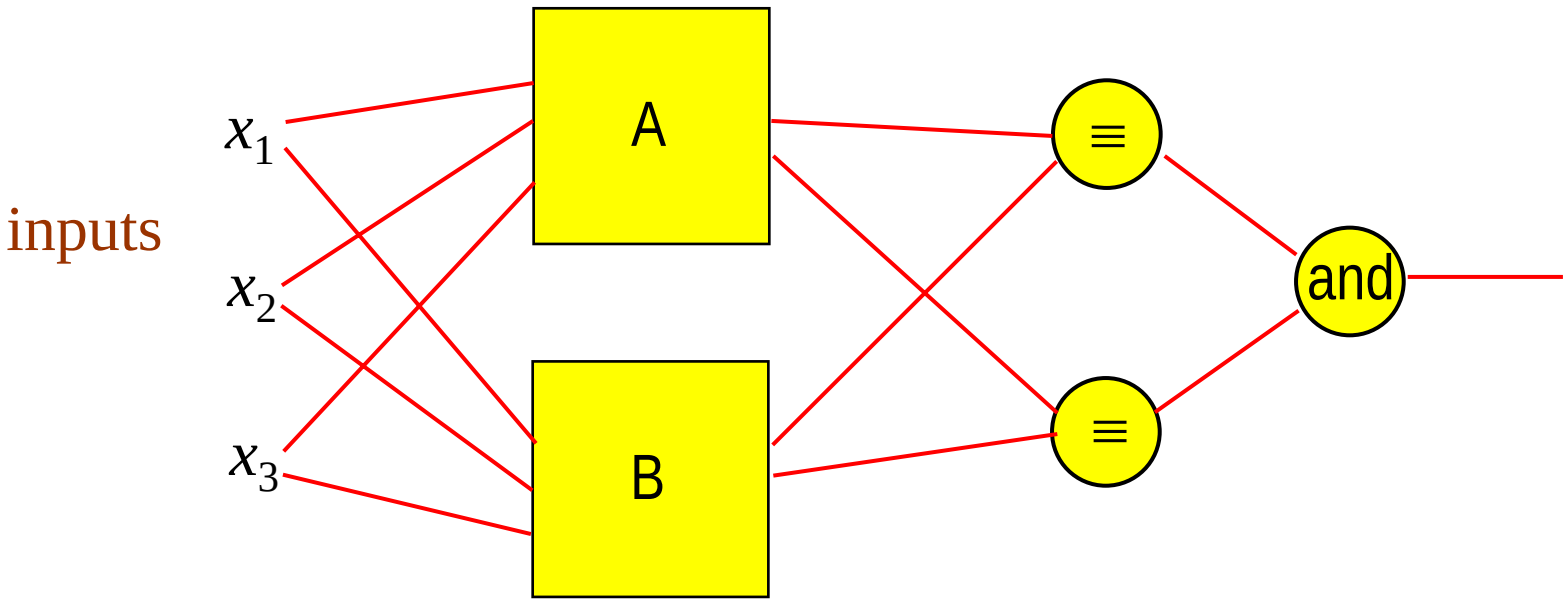
$$\begin{aligned} \min_{x} \quad & v \\ \text{s.t.} \quad & v \geq f(x) + u^k (a - g(x)), k = 0, \ldots, K \end{aligned}$$

Where $u^1, \ldots, u^k$ are the solutions of the first K subproblem duals. The case of an infeasible subproblem requires special treatment.

Logic circuit verification

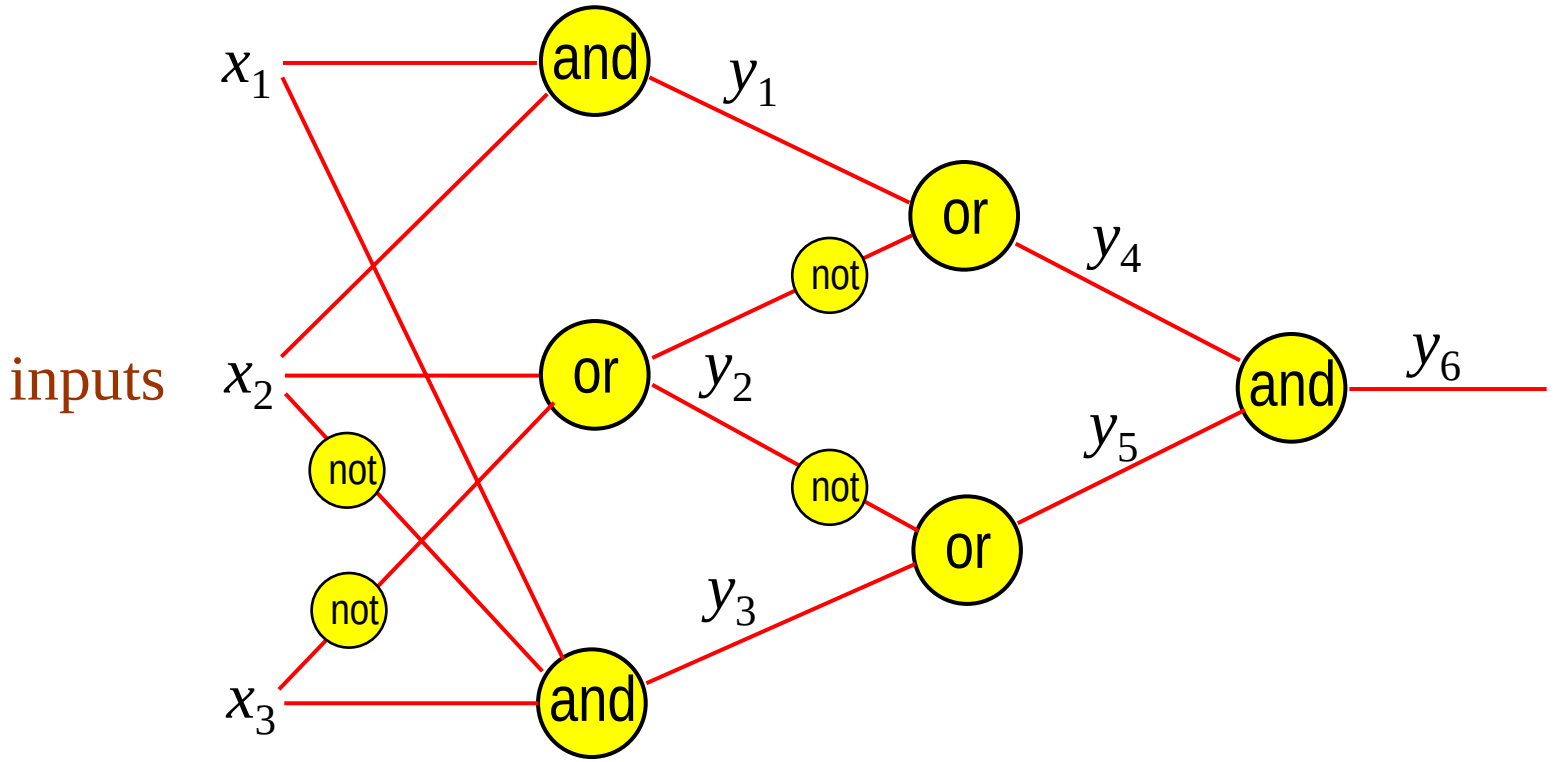
(*JNH, Yan)*

Logic circuits A and B are equivalent when the following circuit is a tautology:



The circuit is a tautology if the minimum output over all 0-1 inputs is 1.

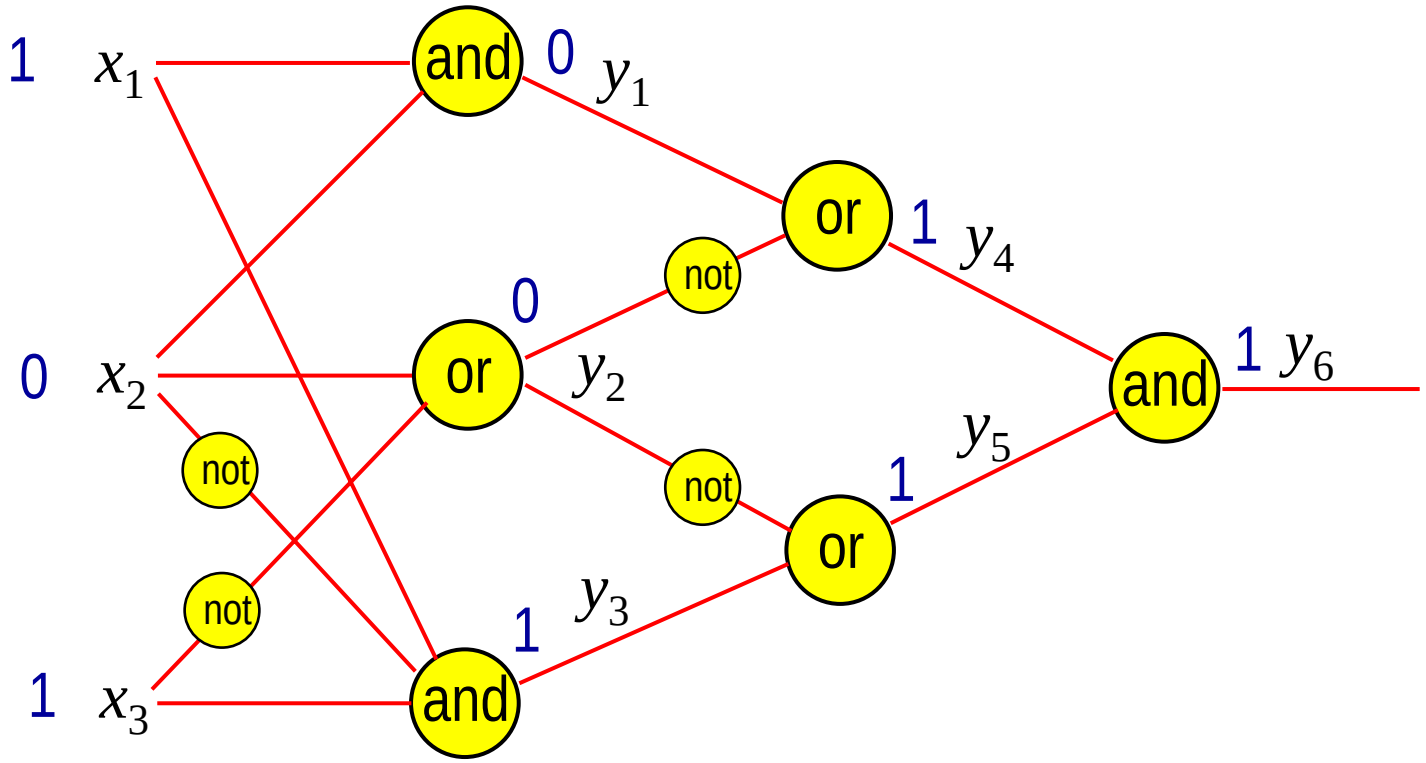
For instance, check whether this circuit is a tautology:



The subproblem is to minimize the output when the input x is fixed to a given value.

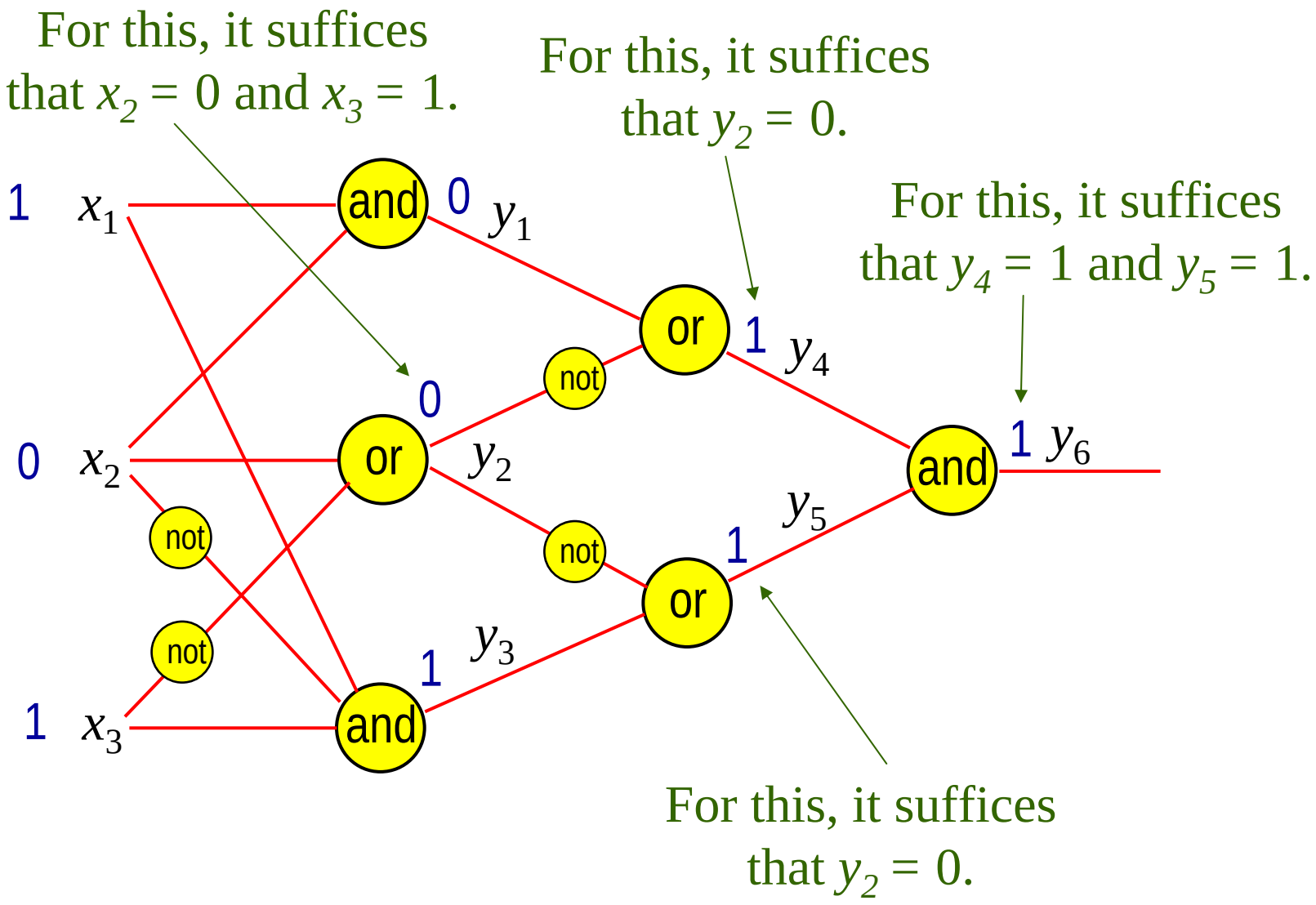
But since x determines the output of the circuit, the subproblem is easy: just compute the output.

For example, let $x = (1,0,1)$.



To construct a Benders cut, identify which subsets of the inputs are sufficient to generate an output of 1.

For instance, $(x_2, x_3) = (0,1)$ suffices.



So, Benders cut is $v \geq \neg x_2 \wedge x_3$

Now solve the master problem

$$\begin{array}{ll} \min & v \\ \text{s.t.} & v \geq \neg x_2 \wedge x_3 \end{array}$$

One solution is $(x_1, x_2, x_3) = (0,0,0)$

This produces output 0, which shows the circuit is not a tautology.

Note: This is actually a case of classical Benders. The subproblem can be written as an LP (a Horn-SAT problem).

Computational results:

Compare with Binary Decision Diagrams (BDDs), state-of-the-art exact method.

- When A and B are equivalent (the circuit is a tautology), BDDs are usually much better.

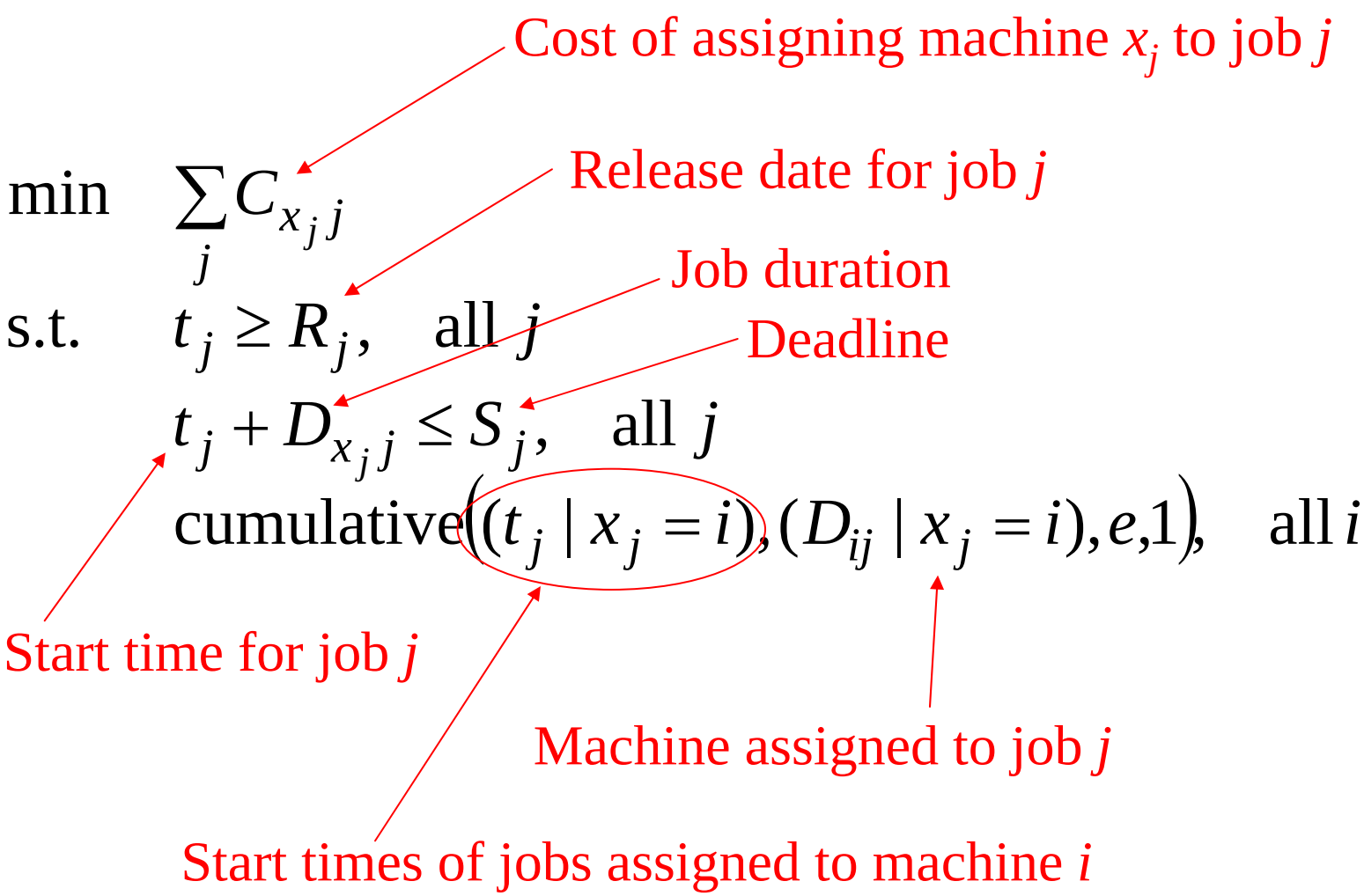
- When A and B are not equivalent (one contains an error), the Benders approach is usually much better.

Machine scheduling

Assign each job to one machine so as to process all jobs at minimum cost. Machines run at different speeds and incur different costs per job. Each job has a release date and a due date.

- In this problem, the master problem assigns jobs to machines. The subproblem schedules jobs assigned to each machine.
- Classical mixed integer programming solves the master problem.
- Constraint programming solves the subproblem, a 1-machine scheduling problem with time windows.
- This provides a general framework for combining mixed integer programming and constraint programming.

A model for the problem:



For a given set of assignments $\bar{x}$ the subproblem is the set of 1-machine problems,

$$\begin{aligned} &\min && 0 \\ &\text{s.t.} && \text{cumulative}\left((t_j \mid \bar{x}_j = i),(D_{ij} \mid \bar{x}_j = i), e, 1\right), \quad \text{all } i \end{aligned}$$

Feasibility of each problem is checked by constraint programming. One or more infeasible problems results in an optimal value ∞. Otherwise the value is zero.

Suppose there is no feasible schedule for machine i. Then jobs $\{ j \mid \bar{x}_j = i \}$ cannot all be assigned to machine i.

Suppose in fact that some subset $J_i(\bar{x})$ of these jobs cannot be assigned to machine i. Then we have a Benders cut

$$v \geq B_{\bar{x}}(x) = \begin{cases} \infty & \text{if } x_j = i \text{ for all } j \in J_i(\bar{x}) \\ 0 & \text{otherwise} \end{cases}$$

Equivalently, just add the constraint

$$x_j \neq i \text{ for some } j \in J_i(\bar{x})$$

This yields the master problem,

$$\begin{aligned} \min \quad & \sum_j C_{x_j j} \\ \text{s.t.} \quad & t_j \geq R_j, \quad \text{all } j \\ & t_j + D_{x_j j} \leq S_j, \quad \text{all } j \\ & x_j \neq i \text{ for some } j \in J_i(x^k), \text{ all } i, \ k = 1, \ldots, K \end{aligned}$$

This problem can be written as a mixed 0-1 problem:

$$\min \quad \sum_{ij} C_{ij} y_{ij}$$

$$\text{s.t.} \quad t_j \geq R_j, \quad \text{all } j$$

$$t_j + \sum_i D_{ij} y_{ij} \leq S_j, \quad \text{all } j$$

$$\sum_i y_{ij} \geq 1, \quad \text{all } j$$

$$\sum_j (1 - y_{ij}) \geq 1, \quad \text{all } i, \quad k = 1, \ldots, K$$

$$x_j^k = i$$

Valid constraint added to improve performance →

$$\sum_j D_{ij} y_{ij} \leq \max_j \{S_j\} - \min_j \{R_j\}, \quad \text{all } i$$

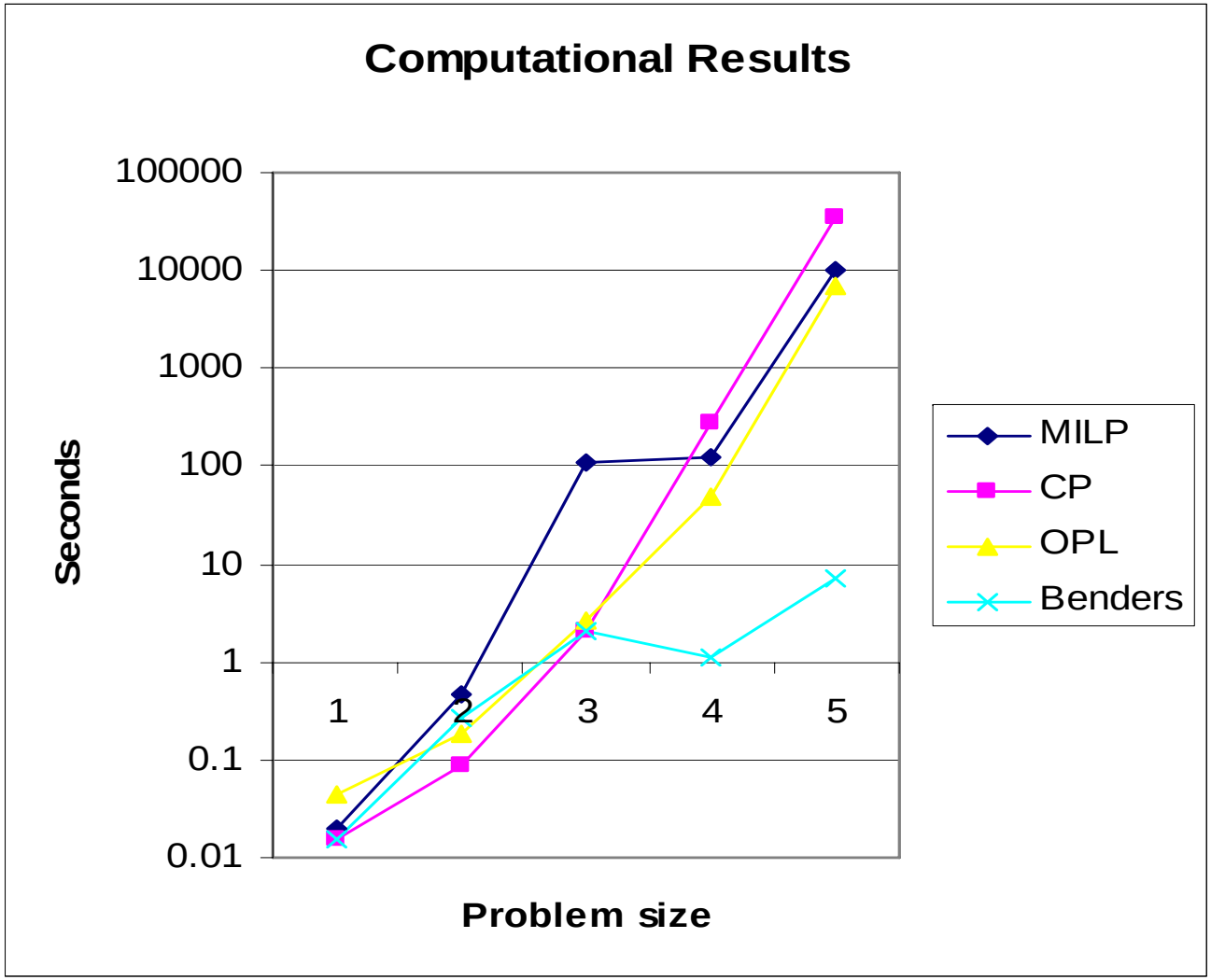
$$y_{ij} \in \{0,1\}$$

Computational Results *(Jain & Grossmann)*

Problem sizes
(jobs, machines)
1 - (3,2)
2 - (7,3)
3 - (12,3)
4 - (15,5)
5 - (20,5)

Each data point represents an average of 2 instances

MILP and CP ran out of memory on 1 of the largest instances



An Enhancement: Branch and Check

(JNH, Thorsteinsson)

- Generate a Benders cut whenever a feasible solution $\bar{x}$ is found in the master problem tree search.
- Keep the cuts (essentially nogoods) in the problem for the remainder of the tree search.
- Solve the master problem only once but continually update it.
- This was applied to the machine scheduling problem described earlier.

Computational results *(Thorsteinsson)*

Computation times in seconds
Problems have 30 jobs, 7 machines.

Problem	Benders	Branch and check
1	16.2	1.2
2	93.7	10.9
3	120.2	1.0
4	37.2	3.0
5	30.2	1.2

Relaxation

Relaxing *all-different*
Relaxing *element*
Relaxing *cycle* (TSP)
Relaxing *cumulative*
Relaxing a disjunction of linear systems
Lagrangean relaxation

Uses of Relaxation

- Solve a relaxation of the problem restriction at each node of the search tree. This provides a bound for the branch-and-bound process.

- In a decomposition approach, place a relaxation of the subproblem into the master problem.

Obtaining a Relaxation

• OR as a well-developed technology for finding polyhedral relaxations for discrete constraints (e.g., cutting planes).

• Relaxations can be developed for global constraints, such as *all-different*, *element*, *cumulative*.

• Disjunctive relaxations are very useful (for disjunctions of linear or nonlinear systems).

Relaxation of *alldifferent*

$$\text{alldiff}(x_1, \ldots, x_n)$$

$$x_j \in \{1, \ldots, n\}$$

Convex hull relaxation, which is the strongest possible linear relaxation *(JNH, Williams & Yan)*:

$$\sum_{j=1}^{n} x_j = \frac{1}{2} n(n+1)$$

$$\sum_{j \in J} x_j \geq \frac{1}{2} |J|(|J|+1), \quad \text{all } J \subseteq \{1, \ldots, n\} \text{ with } |J| < n$$

For $n = 4$:

$$x_1 + x_2 + x_3 + x_4 = 10$$

$$x_1 + x_2 + x_3 \geq 6, \quad x_1 + x_2 + x_4 \geq 6, \quad x_1 + x_3 + x_4 \geq 6, \quad x_2 + x_3 + x_4 \geq 6$$

$$x_1 + x_2 \geq 3, \quad x_1 + x_3 \geq 3, \quad x_1 + x_4 \geq 3, \quad x_2 + x_3 \geq 3, \quad x_2 + x_4 \geq 3, \quad x_3 + x_4 \geq 3$$

$$x_1, x_2, x_3, x_4 \geq 1$$

Relaxation of *element*

To implement variably indexed constant a_y

Replace a_y with z and add constraint $\text{element}(y,(a_1,\ldots,a_n),z)$

Convex hull relaxation of element constraint is simply

$$\min_{j\in D_y}\{a_j\} \le z \le \max_{j\in D_y}\{a_j\}$$

Current domain of y

Relaxation of *element*

To implement variably indexed variable x_{y}

Replace x_{y} with z and add constraint $\text{element}(y,(x_1,\ldots,x_n),z)$ which posts constraint $\bigvee_{j\in D_y}(z=x_j)$

If $0 \le x_j \le m_0$ for each j, there is a simple convex hull relaxation *(JNH):*

$$\sum_{j\in D_y} x_j - \left(|D_y|-1\right)m_0 \le z \le \sum_{j\in D_y} x_j$$

If $0 \le x_j \le m_j$ for each j, another relaxation is

$$\frac{\sum_{j\in D_y} \frac{x_j}{m_j} - |D_y| + 1}{\sum_{j\in D_y} \frac{1}{m_j}} \le z \le \frac{\sum_{j\in D_y} \frac{x_j}{m_j} + |D_y| - 1}{\sum_{j\in D_y} \frac{1}{m_j}}$$

Example:

x_y, where $D_y = \{1,2,3\}$ and

$$0 \le x_1 \le 3$$
$$0 \le x_2 \le 4$$
$$0 \le x_3 \le 5$$

Replace x_y with z and element$(y,(x_1,x_2,x_3),z)$

Relaxation:

$$x_1 + x_2 + x_3 - 10 \le z \le x_1 + x_2 + x_3$$
$$\frac{20}{47}x_1 + \frac{15}{47}x_2 + \frac{12}{47}x_3 - \frac{120}{47} \le z \le \frac{20}{47}x_1 + \frac{15}{47}x_2 + \frac{12}{47}x_3 + \frac{120}{47}$$

Relaxation of *cycle*

Use classical cutting planes for traveling salesman problem:

Distance from city j to city y_j

$$\begin{aligned} \min \quad & \sum_j c_{jy_j} \\ \text{subject to} \quad & \text{cycle}(y_1,\ldots,y_n) \end{aligned}$$

Visit each city exactly once in a single tour

y_j = city immediately following city j

Can also write:

$$\begin{aligned} \min \quad & \sum_j c_{y_j y_{j+1}} \\ \text{subject to} \quad & \text{all-different}(y_1,\ldots,y_n) \end{aligned}$$

y_j = jth city in tour

Begin with 0-1 model and use its continuous relaxation:

$$\min \quad \sum_e c_e x_e$$

$$\text{subject to} \quad \sum_{e \in \delta(i)} x_e = 2, \quad \text{all } i \in V$$

$$\sum_{e \in E(S)} x_e \le |S| - 1, \quad \text{all } S \subset V,\ 2 \le |S| \le |V| - 2$$

$$0 \le x_e \le 1$$

Vertex degree constraints

Subtour elimination constraints

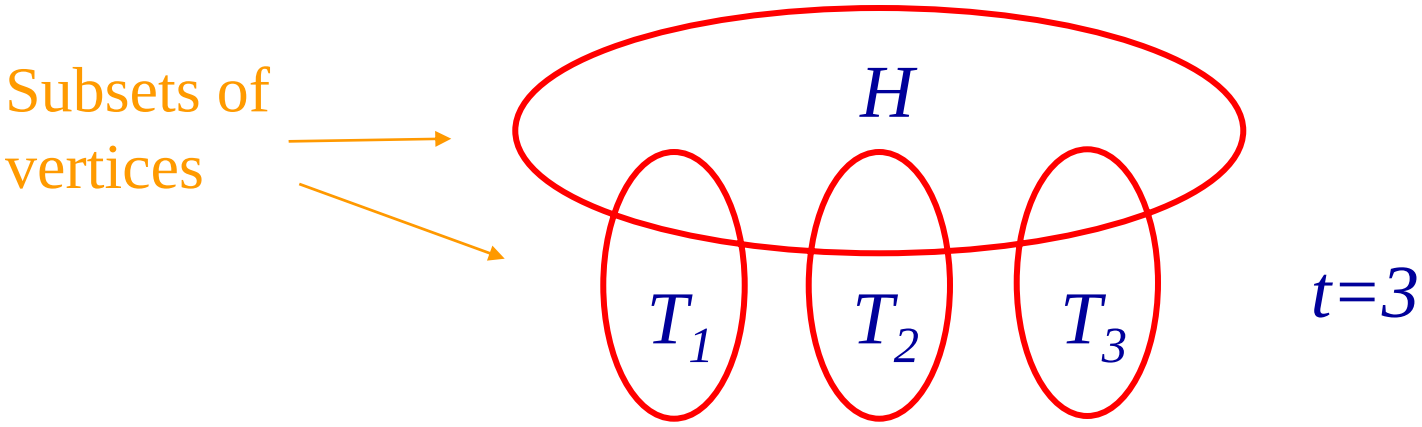
Add *separating* constraints (those violated by current solution of relaxation)

=1 if edge *e* is in the tour, 0 otherwise

$\delta(i)$ = {edges incident to vertex i}

$E(S)$ = {edges with both vertices in S}

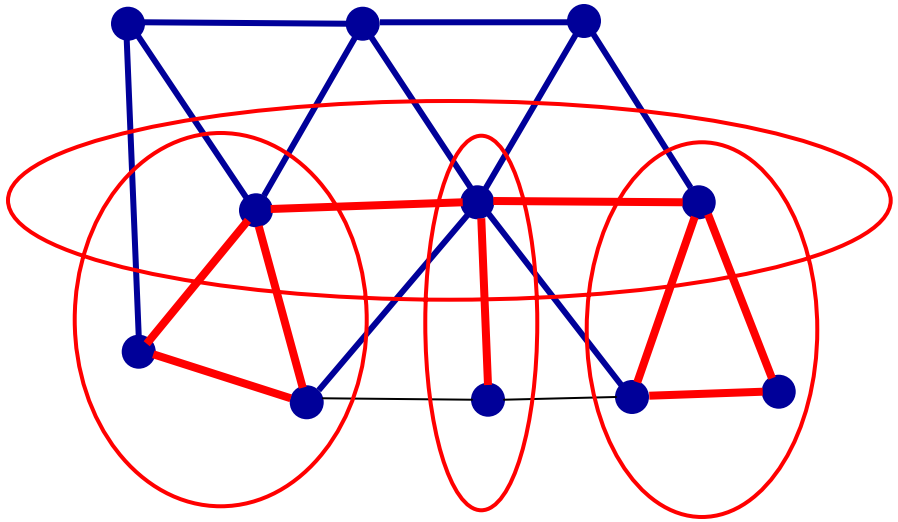
Add more separating cutting planes, such as *comb inequalities (Grötschel, Padberg)*



$$\sum_{e \in E(H)} x_e + \sum_{j=1}^{t} \sum_{e \in E(T_j)} x_e \leq |H| + \sum_{j=1}^{t} |T_j| - \lceil 3t/2 \rceil$$

These inequalities are facet-defining for the convex hull of the feasible set.

Example:



Sum over red edges.

$$\sum_{e} x_e \leq 6$$

• There are polynomial separation algorithms for special classes of comb inequalities.

• In general, one can identify substructures that can be completely analyzed in order to generate valid constraints.

Relaxation of *cumulative*

(JNH, Yan)

$$\text{cumulative}(t, d, r, L)$$

Where $t = (t_1,\ldots,t_n)$ are job start times

$d = (d_1,\ldots,d_n)$ are job durations

$r = (r_1,\ldots,r_n)$ are resource consumption rates

L is maximum total resource consumption rate

$a = (a_1,\ldots,a_n)$ are earliest start times

One can construct a relaxation consisting of the following valid cuts.

If some subset of jobs $\{j_1,\ldots,j_k\}$ are identical (same release time a_0, duration d_0, and resource consumption rate r_0), then

$$t_{j_1} + \cdots + t_{j_k} \geq (P+1)a_0 + \frac{1}{2}P\left[2k - (P+1)Q\right]d_0$$

is a valid cut and is facet-defining if there are no deadlines, where

$$Q = \left\lfloor \frac{L}{r_0} \right\rfloor, \qquad P = \left\lceil \frac{k}{Q} \right\rceil - 1$$

The following cut is valid for any subset of jobs $\{j_1,\ldots,j_k\}$

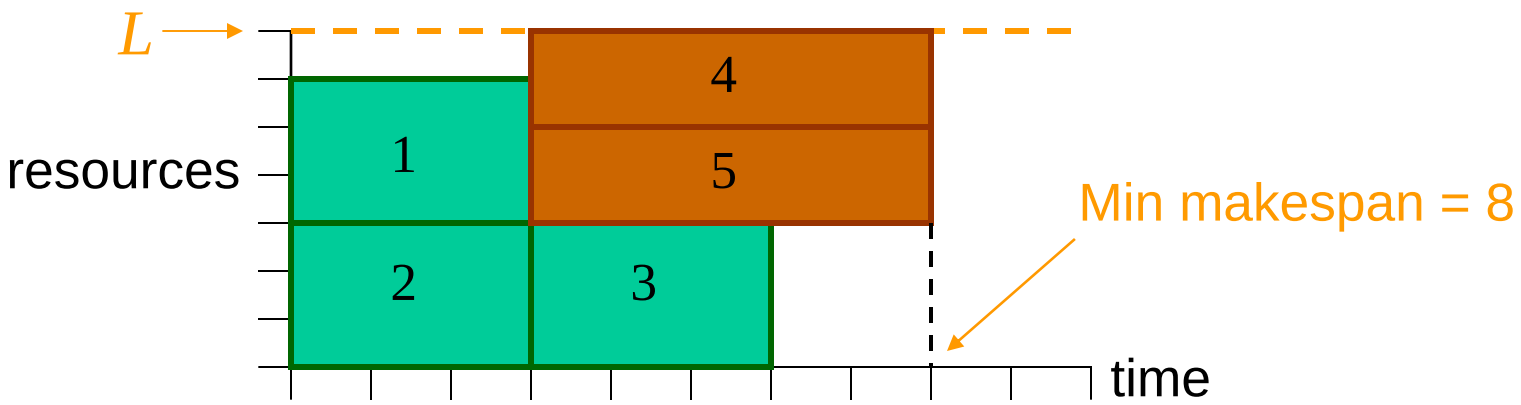
$$t_{j_1} + \cdots + t_{j_k} \geq \sum_{i=1}^{k} \left((k - i + \frac{1}{2}) \frac{r_i}{L} - \frac{1}{2} \right) d_i$$

Where the jobs are ordered by nondecreasing $r_j d_j$.

Analogous cuts can be based on deadlines.

Example:

Consider problem with following minimum makespan solution (all release times = 0):



Relaxation:

$$\begin{aligned}
\min \quad & z \\
\text{s.t.} \quad & z \geq t_1 + 3,\ t_2 + 3,\ t_3 + 3,\ t_4 + 5,\ t_5 + 5 \\
& t_1 + t_2 + t_3 \geq 3 \qquad \leftarrow \text{Facet defining} \\
& t_1 + t_2 + t_3 + t_4 \geq 3\tfrac{5}{14} \\
& t_2 + t_3 + t_4 + t_5 \geq 2\tfrac{4}{7} \\
& t_1 + t_2 + t_3 + t_4 + t_5 \geq 6\tfrac{6}{7} \\
& t_j \geq 0
\end{aligned}$$

Resulting bound:

$$z = \text{makespan} \geq 5.17$$

Relaxing Disjunctions of Linear Systems

$$\bigvee_k \left(A^k x \le b^k \right)$$

(*Element* is a special case.)

Convex hull relaxation *(Balas)*.

$$A^k x^k \le b^k y_k, \quad \text{all } k$$

$$x = \sum_k x^k$$

$$\sum_k y_k = 1$$

$$y_k \ge 0$$

Additional variables needed.

Can be extended to nonlinear systems *(Stubbs & Mehrotra)*

“Big M” relaxation

$$A^k x \le b^k - M^k(1 - y_k), \quad \text{all } k$$

$$\sum_k y_k = 1$$

$$y_k \ge 0$$

Where (taking the max in each row):

$$M_i^k = \max_k \left\{ \max_x \{ A_i^k x \mid A_i^{k'} \le b_i^{k'}, \text{all } k' \ne k \} \right\} - b_i^k$$

This simplifies for a disjunction of inequalities $\bigvee_{k=1}^{K} \left(a^k x \le b_k \right)$ where $0 \le x_j \le m_j$ *(Beaumont)*:

$$\left(\sum_{k=1}^{K} \frac{a^k}{M_k} \right) x \le \sum_{k=1}^{K} \frac{b_k}{M_k} + K - 1$$

where

$$M_k = \sum_j \max\left\{ 0, a_j^k \right\} m_j$$

Example:

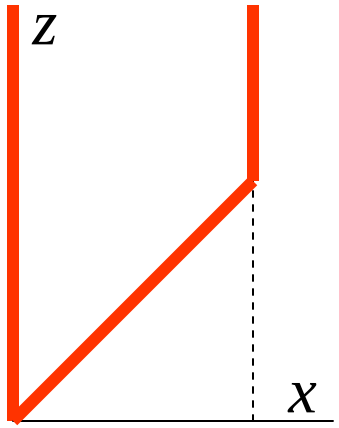
$$\begin{pmatrix}\text{no machine}\\ x=0\end{pmatrix}\vee\begin{pmatrix}\text{small machine}\\ z=50\\ x\le 5\end{pmatrix}\vee\begin{pmatrix}\text{large machine}\\ z=80\\ x\le 10\end{pmatrix}$$

Output of machine

Fixed cost of machine

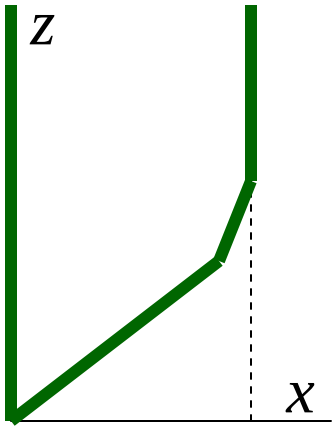
Convex hull relaxation:

$$z \ge 50y_2 + 80y_3$$
$$x \le 5y_2 + 10y_3$$
$$y_2 + y_3 \le 1$$
$$y_2, y_3 \ge 0$$



Big-M relaxation:

$$x \le 10y_2 + 10y_3$$
$$x \le 10 - 5y_2$$
$$x \le 5 + 5y_3$$
$$z \ge 50y_2$$
$$z \ge 80y_3$$
$$y_2 + y_3 \le 1$$
$$y_2, y_3 \ge 0$$



Lagrangean Relaxation

A relaxation in which the hard constraints are moved to the objective function.

$$\begin{array}{ll} \min & f(x) \\ \text{subject to} & g(x) \geq 0 \\ & x \in S \end{array}$$

hard

easy

The relaxation is parameterized by a vector of Langrange multipliers λ.

Dualized constraints

$$\theta(\lambda) = \min_{x \in S} \{ f(x) - \lambda^T g(x) \}$$

A lower bound is obtained by solving the *Lagrangean dual*:

$$\max_{\lambda \geq 0} \{\theta(\lambda)\}$$

Can use *subgradient optimization* to solve the dual.
Exploit the fact that θ(λ) is concave (but nondifferentiable).

A subgradient of θ(λ) is $-g(x^*)$, where x^* solves inner problem.

$$\theta(\lambda) = \min_{x \in S} \{ f(x) - \lambda^T g(x) \}$$

Step k of search is $\lambda^{k+1} = \lambda^k + \alpha_k g(x^*)$

Simplest stepsize is $\alpha_k = \frac{\beta}{k}$

Example:

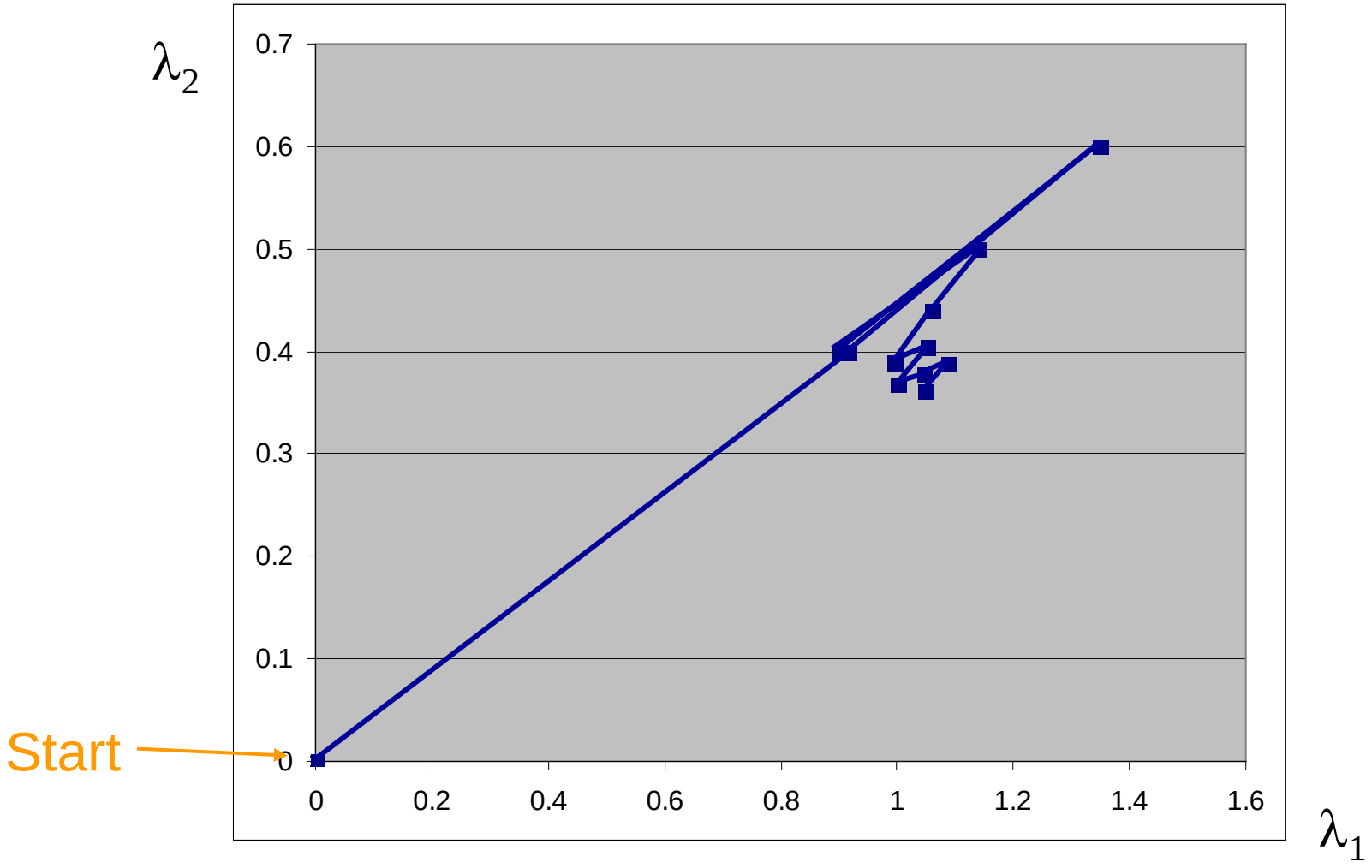
$$
\begin{aligned}
\min \quad & 4x_1 + 5x_2 + 6x_3 \\
\text{subject to} \quad & 3x_1 + 4x_2 + 5x_3 \geq 47 \\
& x_1 + 2x_2 + 3x_3 \geq 24 \qquad \} \text{ hard} \\
& x_1 \geq 2 \\
& x_2 \geq 3 \qquad \} \text{ easy} \\
& x_3 \geq 4 \\
& x \text{ integer}
\end{aligned}
$$

$$
\theta(\lambda) = \min_{\substack{x_1 \geq 2 \\ x_2 \geq 3 \\ x_3 \geq 4}} \left\{ \begin{array}{l} 4x_1 + 5x_2 + 6x_3 \\ + \lambda_1 (47 - 3x_1 - 4x_2 - 5x_3) \\ + \lambda_2 (24 - x_1 - 2x_2 - 3x_3) \end{array} \right\}
$$

Solve by inspection for fixed λ

$$
= \min_{\substack{x_1 \geq 2 \\ x_2 \geq 3 \\ x_3 \geq 4}} \left\{ \begin{array}{l} (4 - 3\lambda_1 - \lambda_2) x_1 + (5 - 4\lambda_1 - 2\lambda_2) x_2 \\ + (6 - 5\lambda_1 - 3\lambda_2) x_3 + 47\lambda_1 + 24\lambda_2 \end{array} \right\}
$$

Subgradient seach in λ-space:



Value of Lagrangean dual = 57.6 < 58 = optimal value

Putting It Together

Elements of a General Scheme
Processing Network Design
Benders Decomposition

Elements of a General Scheme

- Model consists of
  - *declaration window* (variables, initial domains)
  - *relaxation windows* (initialize relaxations & solvers)
  - *constraint windows* (each with its own syntax)
  - *objective function* (optional)
  - *search window* (invokes propagation, branching, relaxation, etc.)
- Basic algorithm searches over problem restrictions, drawing inferences and solving relaxations for each.

Elements of a General Scheme

- Relaxations may include:
  - Constraint store (with domains)
  - Linear programming relaxation, etc.
- The relaxations link the windows.
  - Propagation (e.g., through constraint store).
  - Search decisions (e.g., nonintegral solutions of linear relaxation).

Elements of a General Scheme

- Constraints invoke specialized inference and relaxation procedures that exploit their structure. For example, they
  - Reduce domains (in-domain constraints added to constraint store).
  - Add constraints to original problems (e.g. cutting planes, logical inferences, nogoods)
  - Add cutting planes to linear relaxation (e.g., Gomory cuts).
  - Add specialized relaxations to linear relaxation (e.g., relaxations for *element*, *cumulative*, etc.)

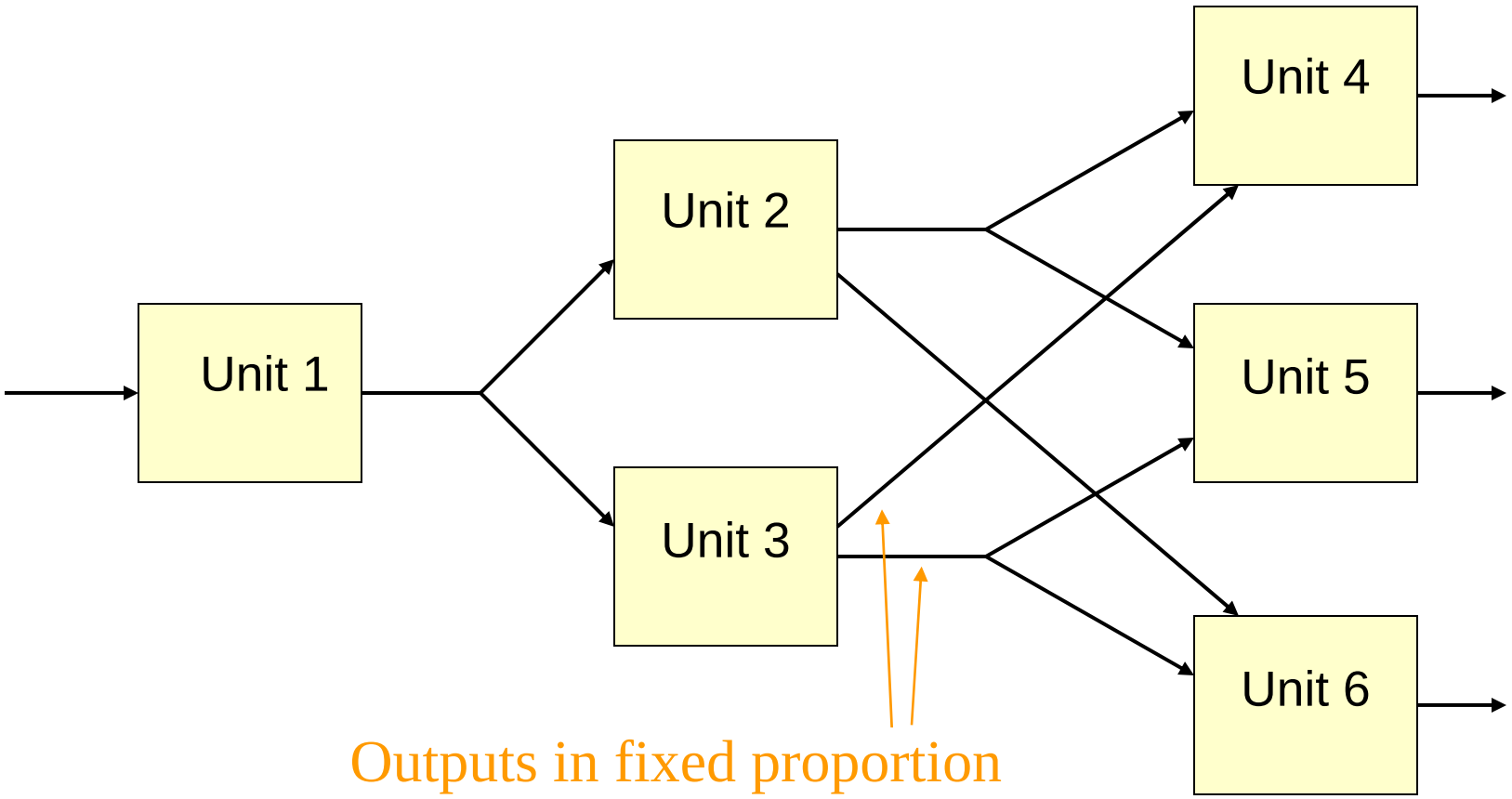
Elements of a General Scheme

- A generic algorithm:
  - Process constraints.
    - Infer new constraints, reduce domains & propagate, generate relaxations.
  - Solve relaxations.
    - Check for empty domains, solve LP, etc.
  - Continue search (recursively).
    - Create new problem restrictions if desired (e.g, new tree branches).
    - Select problem restriction to explore next (e.g., backtrack or move deeper in the tree).

Example: Processing Network Design

- Find optimal design of processing network.
  - A “superstructure” (largest possible network) is given, but not all processing units are needed.
  - Internal units generate negative profit.
  - Output units generate positive profit.
  - Installation of units incurs fixed costs.
  - Objective is to maximize net profit.

Sample Processing Superstructure



Declaration Window

$u_i \in [0,c_i]$	flow through unit i
$x_{ij} \in [0,c_{ij}]$	flow on arc (i,j)
$z_i \in [0,\infty]$	fixed cost of unit i
$y_i \in D_i = \{\text{true,false}\}$	presence or absence of unit i

Objective Function Window

$$\max \sum_i (r_i u_i - z_i)$$

Net revenue generated by unit *i* per unit flow

Relaxation Window

Type: **Constraint store**, consisting of variable domains.

Objective function: None.

Solver: None.

Relaxation Window

Type: **Linear programming.**

Objective function: Same as original problem.

Solver: LP solver.

Constraint Window

Type: **Linear (in)equalities.**

$\boldsymbol{Ax + Bu = b}$ (flow balance equations)

Inference: Bounds consistency maintenance.

Relaxation: Add reduced bounds to constraint store.

Relaxation: Add equations to LP relaxation.

Constraint Window

Type: **Disjunction of linear inequalities.**

$$\begin{pmatrix} y_i \\ z_i \geq d_i \end{pmatrix} \vee \begin{pmatrix} \neg y_i \\ u_i \leq 0 \end{pmatrix}$$

Inference: None.

Relaxation: Add Beaumont's projected big-M relaxation to LP.

Constraint Window

Type: **Propositional logic.**

Don't-be-stupid constraints:

$y_1 \rightarrow (y_2 \vee y_3)$ $\quad y_3 \rightarrow y_4$

$y_2 \rightarrow y_1$ $\quad y_3 \rightarrow (y_5 \vee y_6)$

$y_2 \rightarrow (y_4 \vee y_5)$ $\quad y_4 \rightarrow (y_2 \vee y_3)$

$y_2 \rightarrow y_6$ $\quad y_5 \rightarrow (y_2 \vee y_3)$

$y_3 \rightarrow y_1$ $\quad y_6 \rightarrow (y_2 \vee y_3)$

Inference: Resolution (add resolvents to constraint set).

Relaxation: Add reduced domains of y_i's to constraint store.

Relaxation (optional): Add 0-1 inequalities representing propositions to LP.

Search Window

Procedure BandBsearch(*P*,*R*,*S*,NetBranch)
(canned branch & bound search using NetBranch as branching rule)

User-Defined Window

Procedure NetBranch(*P*,*R*,*S*,*i*)

Let i be a unit for which $u_i > 0$ and $z_i < d_i$.
If $i = 1$ then create P' from P by letting $D_i = \{T\}$
and return P'.
If $i = 2$ then create P' from P by letting $D_i = \{F\}$
and return P'.

Benders Decomposition

- Benders is a special case of the general framework.
  - The Benders subproblems are problem restrictions over which the search is conducted.
  - Benders cuts are generated constraints.
  - The Master problem is the relaxation.
  - The solution of the relaxation determines which subproblem to solve next.

Other OR Techniques of Interest to CP

- Column generation for LP, branch-and-price *(when there are many variables).*
- Reduced-cost variable fixing *(recently used for cost-based domain filtering).*
- Nonlinear programming *(well-developed technology)*
  - Active set methods (generalized reduced gradient)
  - Variable metric methods, conjugate gradient methods *(unconstrained problems)*
  - Sequential quadratic programming, outer approximation
  - Interior point methods (for LP and NLP).

Other OR Techniques of Interest to CP

- Multicriteria optimization *(compute pareto-optimal solutions, etc.)*
- Optimal control
  - Dynamic programming *(recursive optimization)*
  - Nonserial dynamic programming *(useful when dependency graph has small induced width)*
  - Calculus of variations, Pontryagin maximum principle *(for continuous problems)*

Other OR Techniques of Interest to CP

- Stochastic methods *(use probabilistic information).*
  - Stochastic dynamic programming, Markov decision models *(optimal control under uncertainty).*
  - Adaptive control *(optimal control + learning)*
  - Stochastic linear programming *(optimization over scenarios).*
  - Queuing.
- Approximation algorithms *(theoretical bounds on accuracy)*
- Heuristic methods *(huge literature).*

Surveys/Tutorials on Hybrid Methods

• A. Bockmayr and J. Hooker, Constraint programming, in K. Aardal, G. Nemhauser and R. Weismantel, eds., *Handbook of Discrete Optimization*, North-Holland, to appear.

• S. Heipcke, *Combined Modelling and Problem Solving in Mathematical Programming and Constraint Programming*, PhD thesis, University of Buckingham, 1999.

• J. Hooker, Logic, optimization and constraint programming, *INFORMS Journal on Computing*, to appear, also at http://ba.gsia.cmu.edu/jnh.

• J. Hooker, *Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction*, Wiley, 2000.

• M. Milano, Integration of OR and AI constraint-based techniques for combinatorial optimization, http://www-lia.deis.unibo.it/Staff/MichelaMilano/tutorialIJCAI2001.pdf

• H. P. Williams and J. M. Wilson, Connections between integer linear programming and constraint logic programming--An overview and introduction to the cluster of articles, *INFORMS Journal on Computing* **10** (1998) 261-264.