Merging Constraint Programming Techniques with Mathematical Programming

Tutorial

John Hooker
Carnegie Mellon University

INFORMS 02
November 2002

This tutorial will available at

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

g

Why Integrate CP and MP?

Search/Inference Duality

Decomposition

Relaxation

Putting It Together

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),
          solution of relaxation & 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.
- CP constraints can be associated with relaxations as well as "filtering" algorithms that eliminate infeasible values of variables.
  - 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

Branch and Infer

Discrete Lot Sizing

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

$$
\begin{aligned}
&5x_1 + 8x_2 + 4x_3 \le z \\
&3x_1 + 5x_2 + 2x_3 \ge 30 \\
&\text{all-different}\{x_1, x_2, x_3\} \\
&x_j \in \{1, \ldots, 4\}
\end{aligned}
$$

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

Global Constraints

All-different is a *global constraint*.

- It represents a set of constraints with special structure.
- This allows a special-purpose inference procedure to be applied.
- The modeler recognizes the structure.

Domain Reduction

Domain reduction is a special type of logical inference method.

- It infers that each variable can take only certain values.
- That is, it reduces the *domains* of the variables.
- Ideally it maintains *hyperarc consistency*: every value in any given domain is part of some feasible solution.

Bounds Consistency

Bounds consistency means that the minimum and maximum element of any given domain are part of some feasible solution.

It is weaker than hyperarc consistency but easier to maintain.

Constraint Propagation

- Domains reduced by one constraint are passed to next constraint and reduced further.
- *Constraint store* (set of domains) allows communication among constraints.
- Keep cycling through domain reduction procedures until fixed point is reached.
- Do this at every node of a branching tree.

Domain reduction for inequalities

- *Bounds propagation* on $\begin{array}{l} 5x_1 + 8x_2 + 4x_3 \le z \\ 3x_1 + 5x_2 + 2x_3 \ge 30 \end{array}$

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

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

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:

```
1 2
1 2
1 2 3 4
```

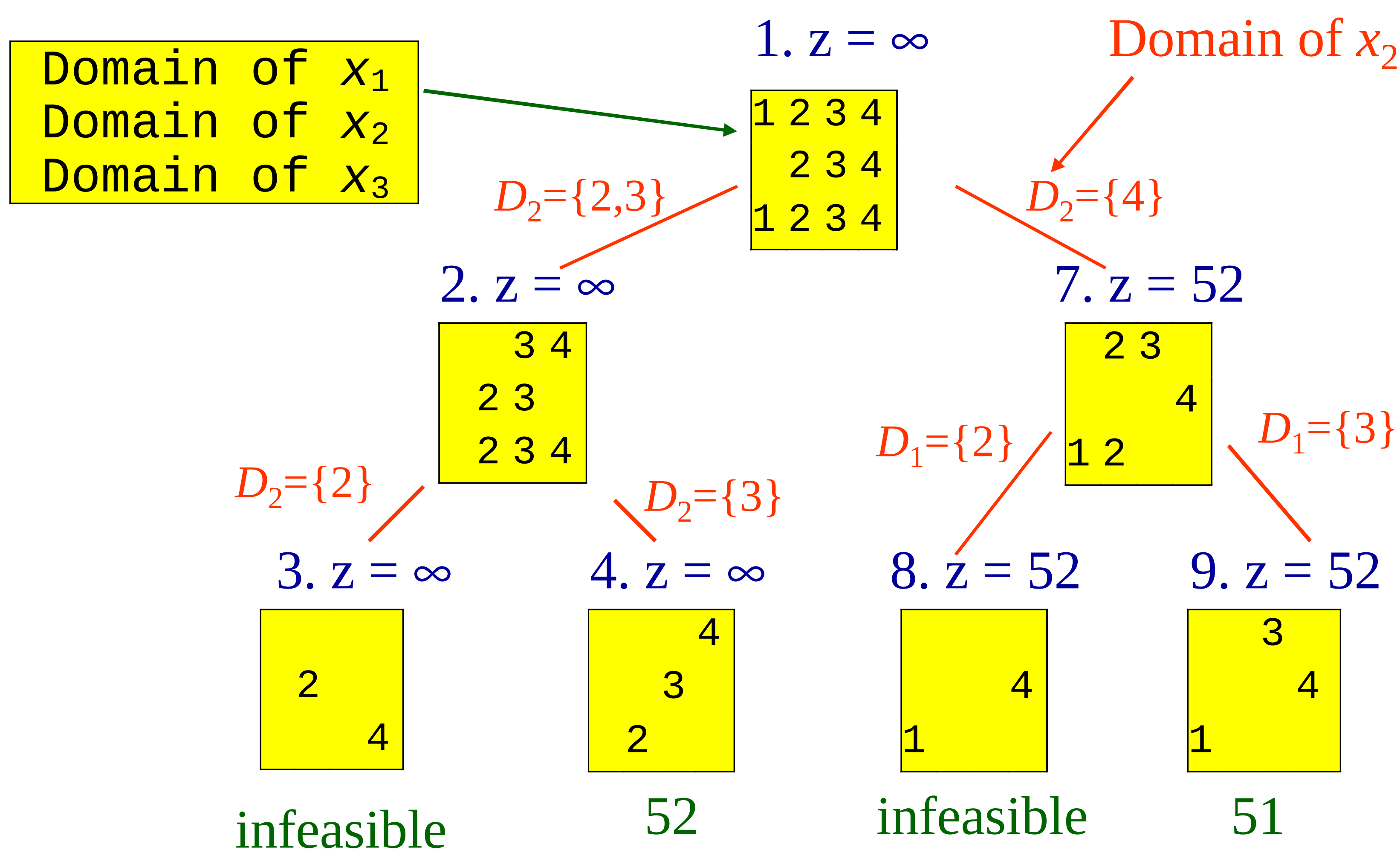
Domain of x_1

Domain of x_2

Domain of x_3

Then one can reduce the domains:

```
1 2
1 2
    3 4
```



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

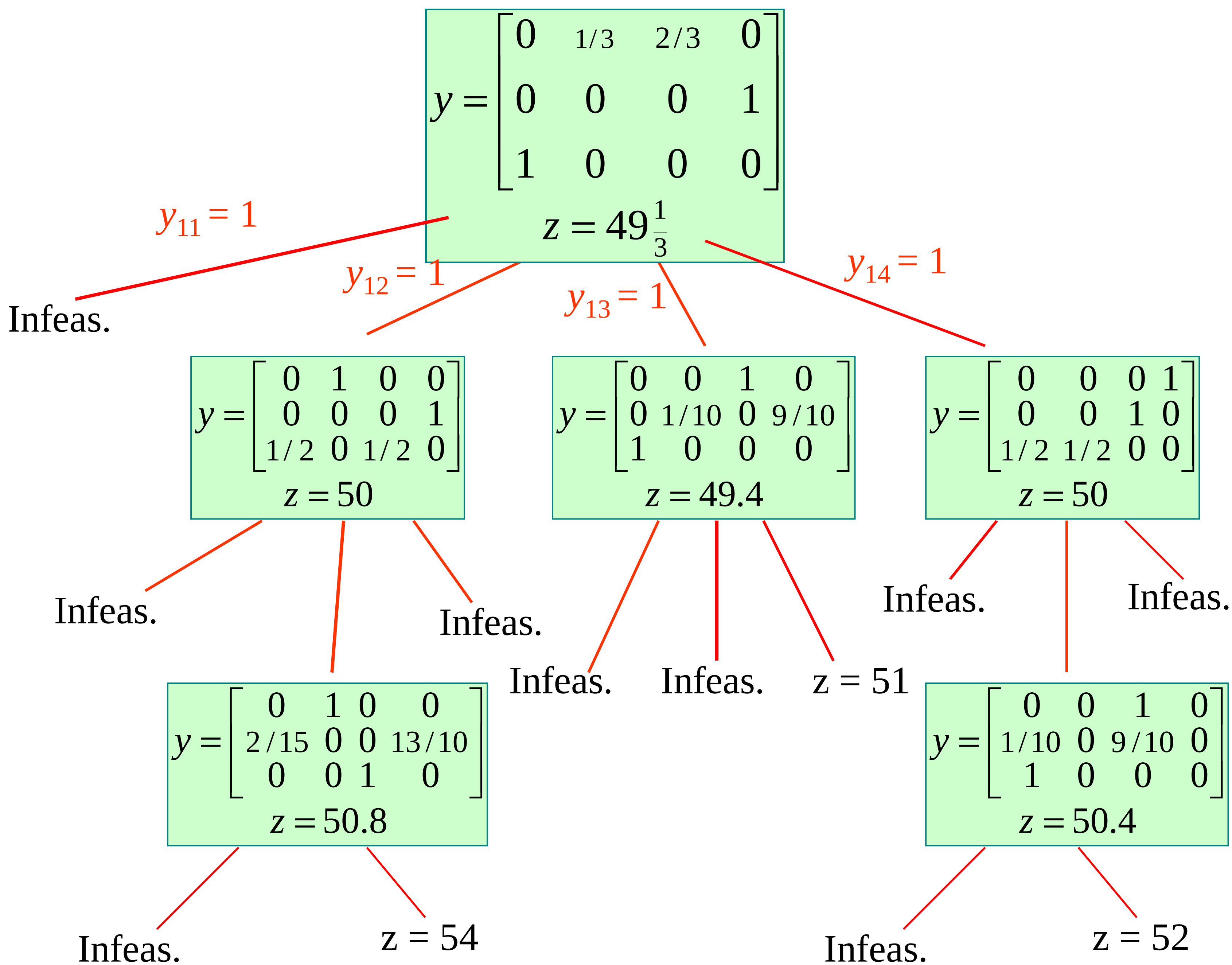
Solve as an integer programming problem

Search: Branch on variables with fractional values in solution of continuous relaxation.
Inference: Solve LP relaxation for lower bound. (Generate cutting planes—not illustrated here).

Rewrite problem using integer programming model:

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

$$\begin{array}{ll}
\text{min} & 5x_1 + 8x_2 + 4x_3 \\
\text{subject to} & 3x_1 + 5x_2 + 2x_3 \geq 30 \\
 & x_i = \sum_{j=1}^{4} j y_{ij}, \quad i = 1,2,3 \\
 & \sum_{j=1}^{4} y_{ij} = 1, \quad i = 1,2,3 \\
 & \sum_{i=1}^{3} y_{ij} \leq 1, \quad j = 1,2,3,4 \\
 & y_{ij} \in \{0,1\}, \quad \text{all } i, j
\end{array}$$



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

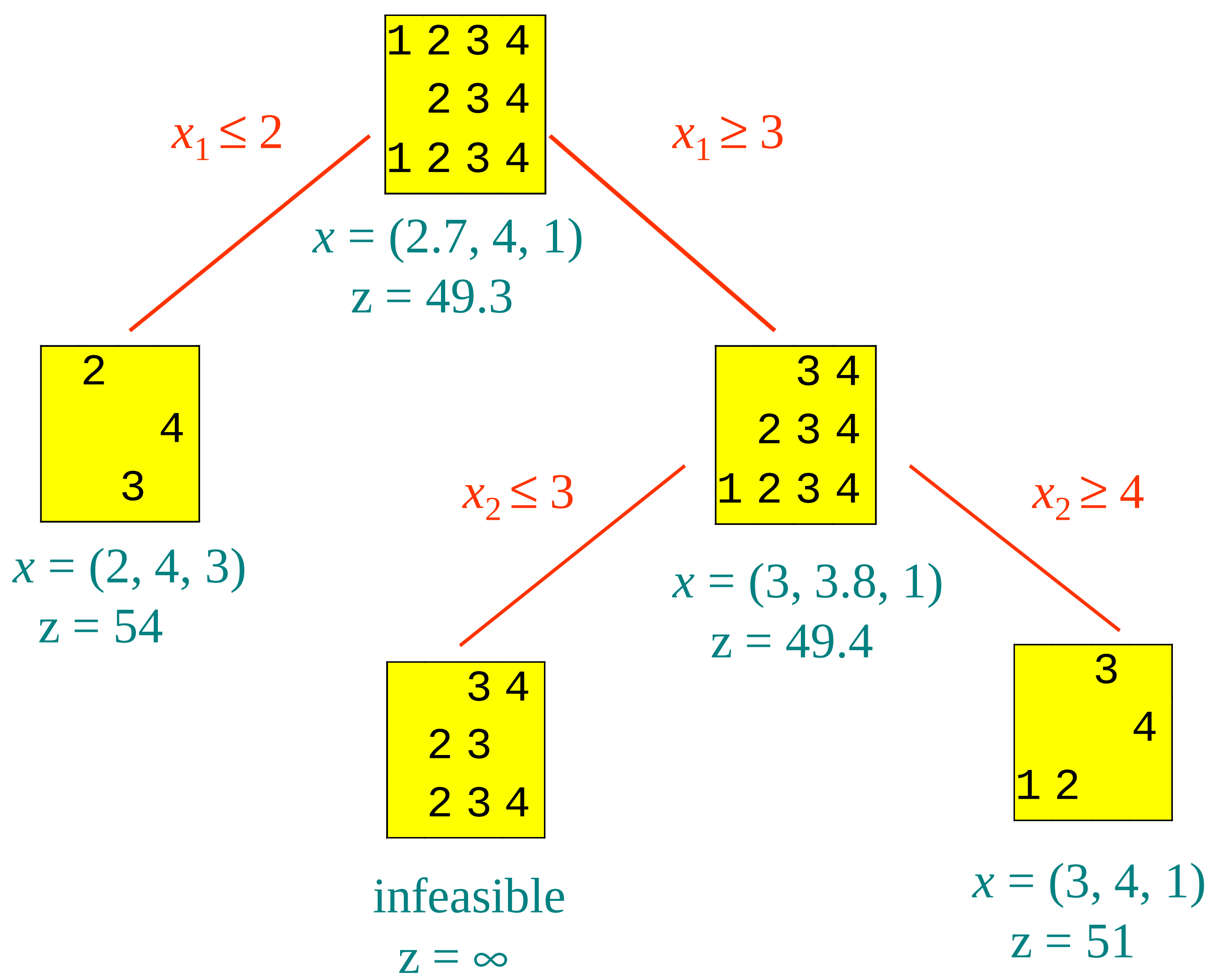
Solve using a hybrid approach

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:

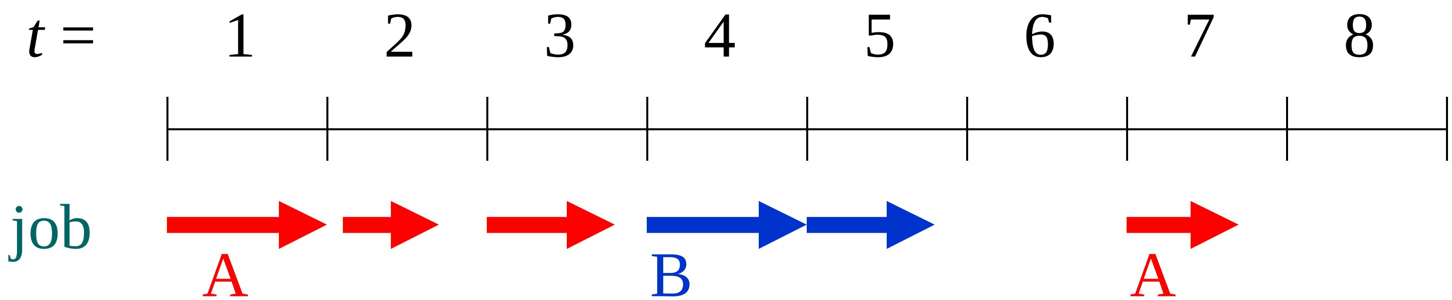
- Use bounds propagation.
- Maintain hyperarc consistency for all-different.
- Get bounds by solving LP relaxation.


1 2 3 4
2 3 4
1 2 3 4
x = (2.7, 4, 1)
z = 49.3
x1 ≤ 2
x1 ≥ 3
2
4
3
x = (2, 4, 3)
z = 54
3 4
2 3 4
1 2 3 4
x = (3, 3.8, 1)
z = 49.4
x2 ≤ 3
x2 ≥ 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)

$$
\begin{aligned}
\min \quad & \sum_{t,i} \left(h_{it} s_{it} + \sum_{j \neq i} q_{ij} \delta_{ijt} \right) \\
\text{s.t.} \quad & s_{i,t-1} + x_{it} = d_{it} + s_{it}, \quad \text{all } i,t \\
& z_{it} \geq y_{it} - y_{i,t-1}, \quad \text{all } i,t \\
& z_{it} \leq y_{it}, \quad \text{all } i,t \\
& z_{it} \leq 1 - y_{i,t-1}, \quad \text{all } i,t \\
& \delta_{ijt} \geq y_{i,t-1} + y_{jt} - 1, \quad \text{all } i,t \\
& \delta_{ijt} \geq y_{i,t-1}, \quad \text{all } i,t \\
& \delta_{ijt} \leq y_{jt}, \quad \text{all } i,t \\
& x_{it} \leq C y_{it}, \quad \text{all } i,t \\
& \sum_{i} y_{it} = 1, \quad \text{all } t \\
& y_{it}, z_{it}, \delta_{ijt} \in \{0,1\} \\
& x_{it}, s_{it} \geq 0
\end{aligned}
$$

Modeling variable indices with *element*

To implement variably indexed constant a_y

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

To implement variably indexed variable x_y

Replace x_y with z and add constraint $\text{element}\big(y, (x_1, \ldots, x_n), z\big)$
which posts the constraint $z = x_y$.

There are straightforward filtering algorithms for *element*.

Hybrid model

total inventory + changeover cost

stock level

changeover cost

daily production

$$
\begin{aligned}
\min \quad & \sum_t (u_t + v_t) \\
\text{s.t.} \quad & u_t \ge \sum_i h_i s_{it}, \ \text{all } t \\
& v_t \ge q_{y_{t-1} y_t}, \ \text{all } t \\
& s_{i,t-1} + x_{it} = d_{it} + s_{it}, \ \text{all } i,t \\
& 0 \le x_{it} \le C, \ s_{it} \ge 0, \ \text{all } i,t \\
& (y_t \ne i) \rightarrow (x_{it} = 0), \ \text{all } i,t
\end{aligned}
$$

inventory balance

To create relaxation:

Put into relaxation

Generate inequalities to put into relaxation (to be discussed)

$$
\begin{aligned}
\min \quad & \sum_t (u_t + v_t) \\
\text{s.t.} \quad & u_t \ge \sum_i h_i s_{it}, \ \text{all } t \\
& v_t \ge q_{y_{t-1} y_t}, \ \text{all } t \\
& s_{i,t-1} + x_{it} = d_{it} + s_{it}, \ \text{all } i,t \\
& 0 \le x_{it} \le C, \ s_{it} \ge 0, \ \text{all } i,t \\
& (y_t \ne i) \rightarrow (x_{it} = 0), \ \text{all } i,t
\end{aligned}
$$

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 Behind Benders Decomposition
Logic Circuit Verification
Machine Scheduling

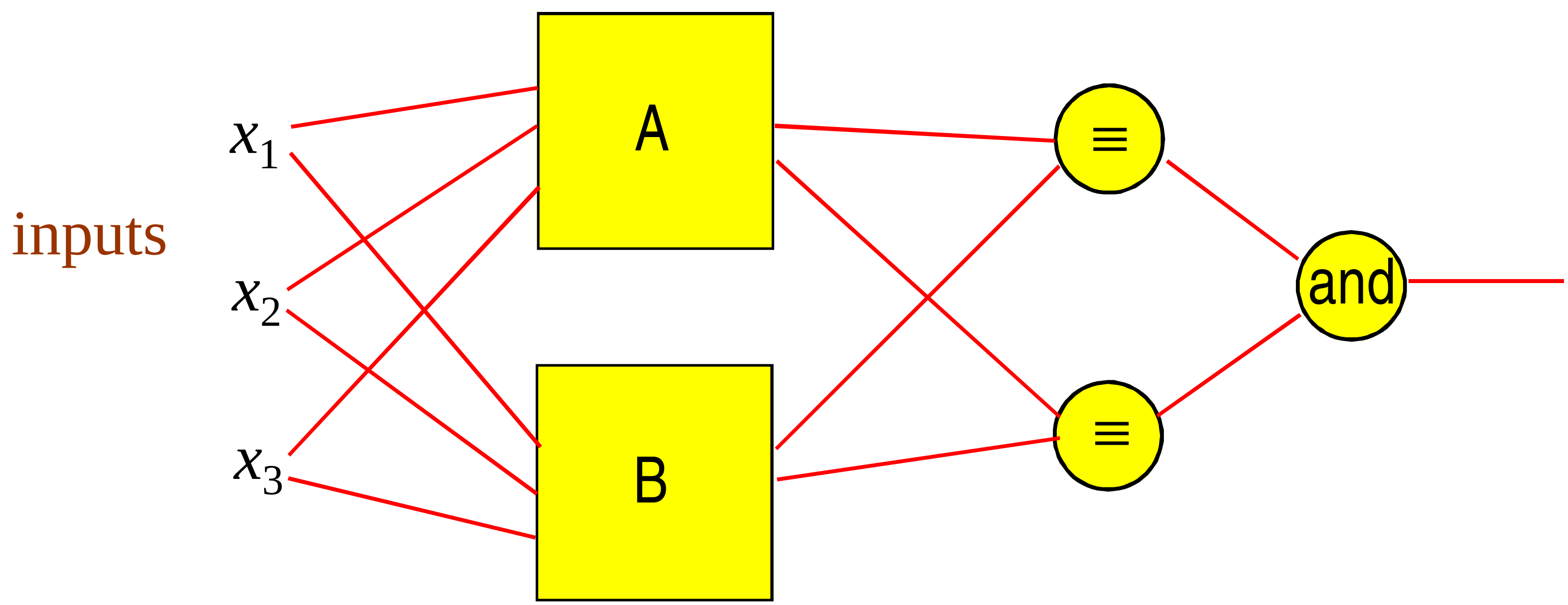
Idea Behind Benders Decomposition
“Learn from one’s mistakes.”

- 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 *Benders cuts* or “nogoods.”
- Add the Benders cut to the master problem to require next solution to be better than last, and re-solve.
- Can also be viewed as projecting problem onto primary variables.

Logic circuit verification

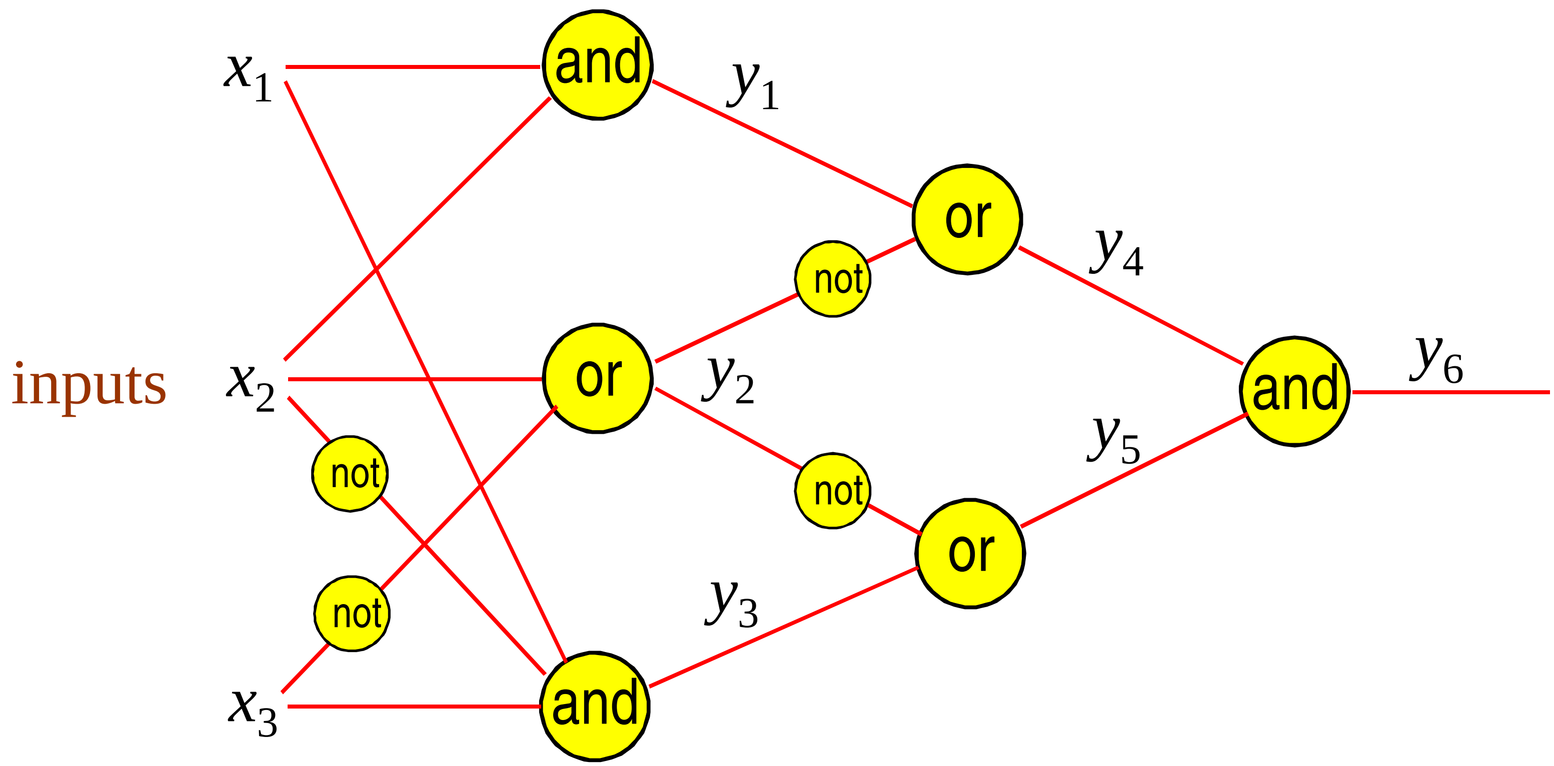
(JNH, Yan)

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



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

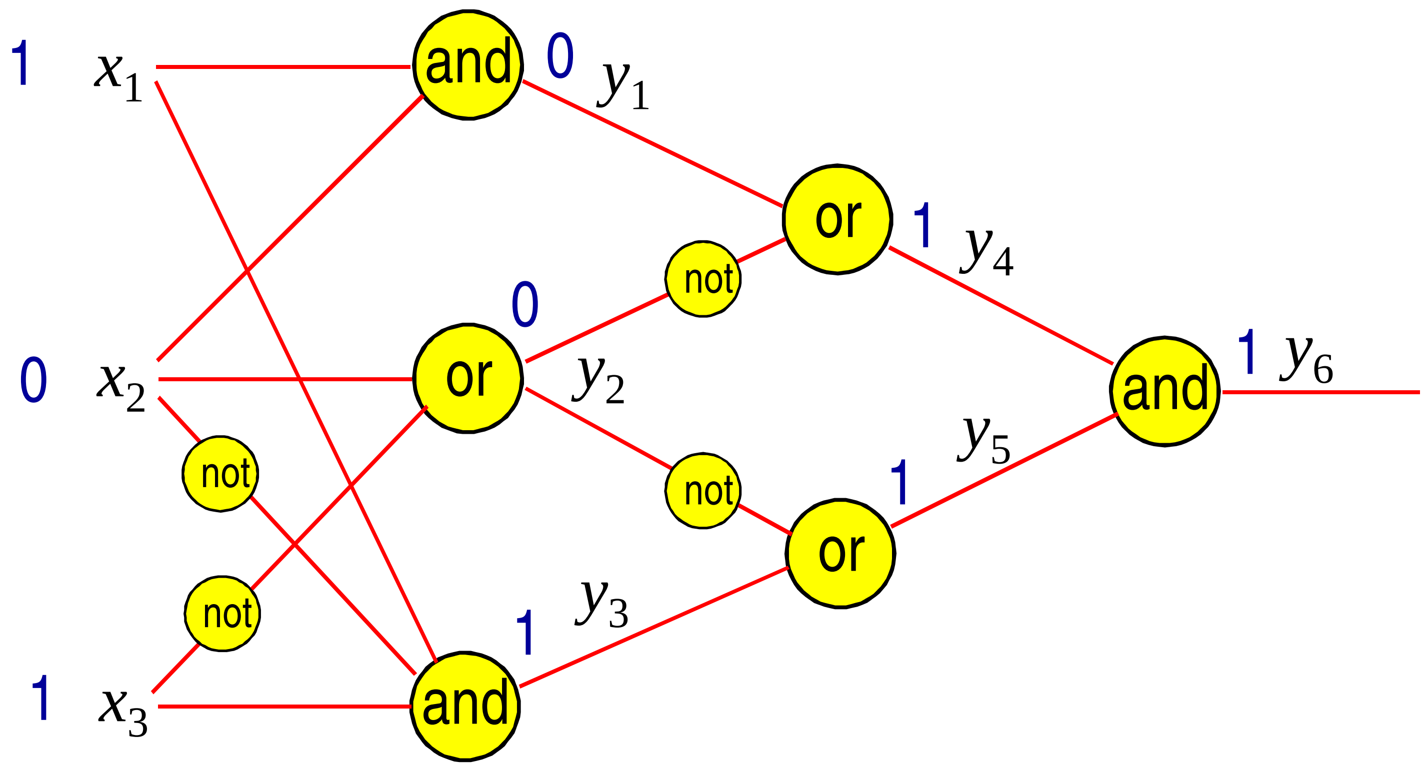
For instance, check whether this circuit is a tautology:



The subproblem is to find whether the output can be 0 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 $x = (1,0,1)$.



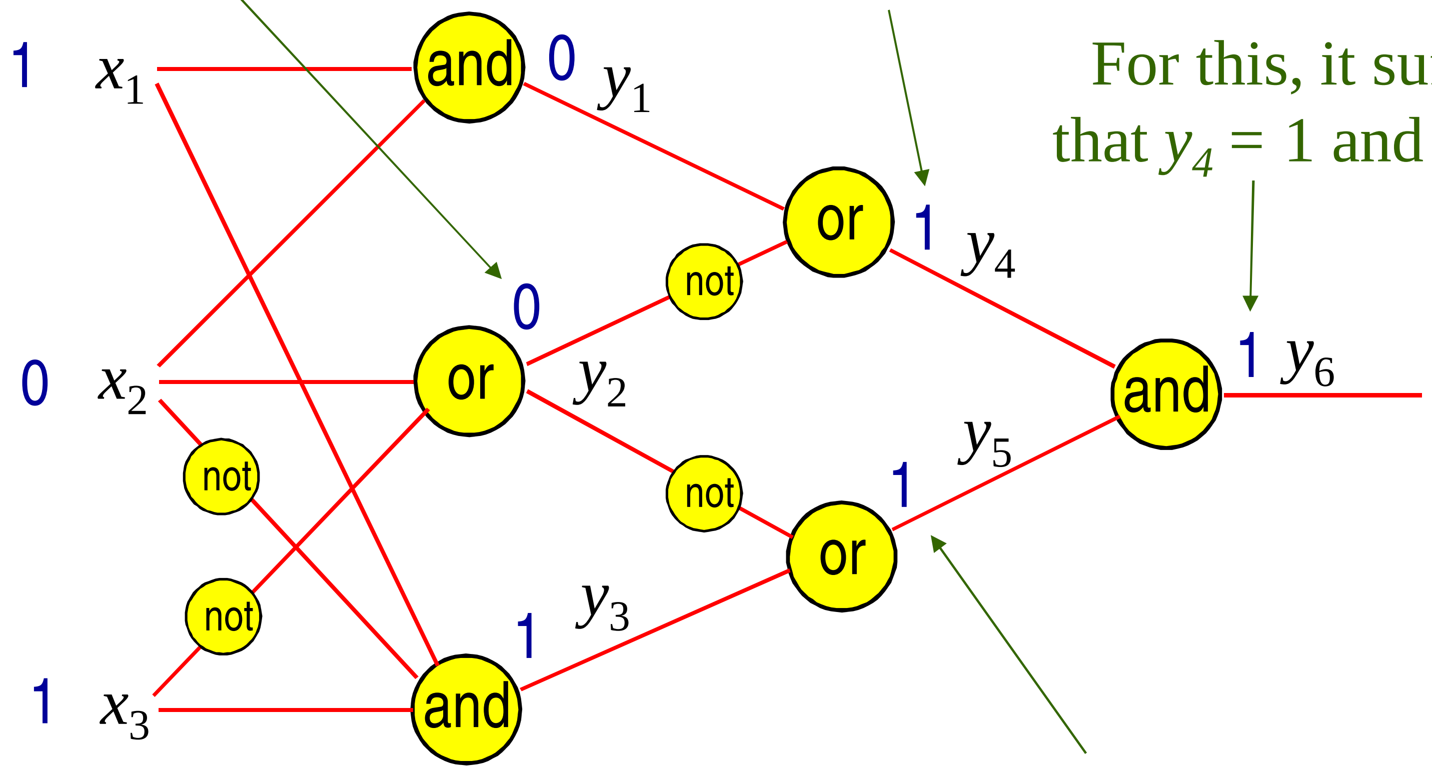
To construct a Benders cut, identify which subsets of the inputs are sufficient to generate an output of 1.

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

For this, it suffices that $x_2 = 0$ and $x_3 = 1$.

For this, it suffices that $y_2 = 0$.

For this, it suffices that $y_4 = 1$ and $y_5 = 1$.



For this, it suffices that $y_2 = 0$.

So, Benders cut is $x_2 \vee \neg x_3$

Now solve the master problem

$$x_2 \vee \neg x_3$$

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).

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.

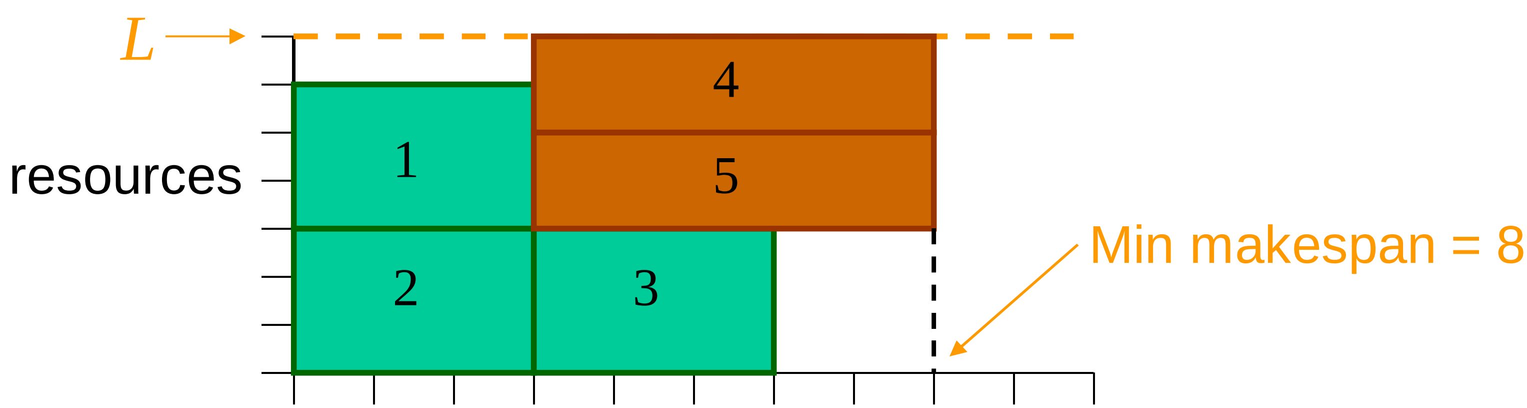
- Classical mixed integer programming solves the master problem.

- 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.

Modeling resource-constrained scheduling with *cumulative*

Jobs 1,2,3 consume 3 units of resources.
Jobs 4,5 consume 2 units.
Maximum L = 7 units of resources available.



$t_1 = t_2 = 0$ $\quad$ $t_3 = t_4 = t_5 = 0$

$$\text{cumulative}\big((t_1,\ldots,t_5),(3,3,3,5,5),(3,3,3,2,2),7\big)$$

Start times $\quad$ Durations $\quad$ Resource consumption $\quad$ L

A model for the machine scheduling problem:

Cost of assigning machine x_j to job j

Release date for job j

Job duration

Deadline

$$
\begin{aligned}
\text{min} \quad & \sum_j C_{x_j j} \\
\text{s.t.} \quad & t_j \ge R_j, \quad \text{all } j \\
& t_j + D_{x_j j} \le S_j, \quad \text{all } j \\
& \text{cumulative}\big((t_j \mid x_j = i), (D_{ij} \mid x_j = i), e, 1\big), \quad \text{all } i
\end{aligned}
$$

Start time for job j

Machine assigned to job j

Resource consumption = 1 for each job

Start times of jobs assigned to machine i

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

$$\text{cumulative}\big((t_j \mid \bar{x}_j = i), (D_{ij} \mid \bar{x}_j = i), e, 1\big), \quad \text{all } i$$

Feasibility of each problem is checked by constraint programming.

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

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

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:

$$
\begin{aligned}
\min \quad & \sum_{ij} C_{ij} y_{ij} \\
\text{s.t.} \quad & t_j \ge R_j, \quad \text{all } j \\
& t_j + \sum_i D_{ij} y_{ij} \le S_j, \quad \text{all } j \\
& \sum_i y_{ij} \ge 1, \quad \text{all } j \\
& \sum_{\substack{j \\ x_j^k = i}} (1 - y_{ij}) \ge 1, \quad \text{all } i, \; k = 1, \ldots, K \\
& \sum_j D_{ij} y_{ij} \le \max_j \{S_j\} - \min_j \{R_j\}, \quad \text{all } i \\
& y_{ij} \in \{0,1\}
\end{aligned}
$$

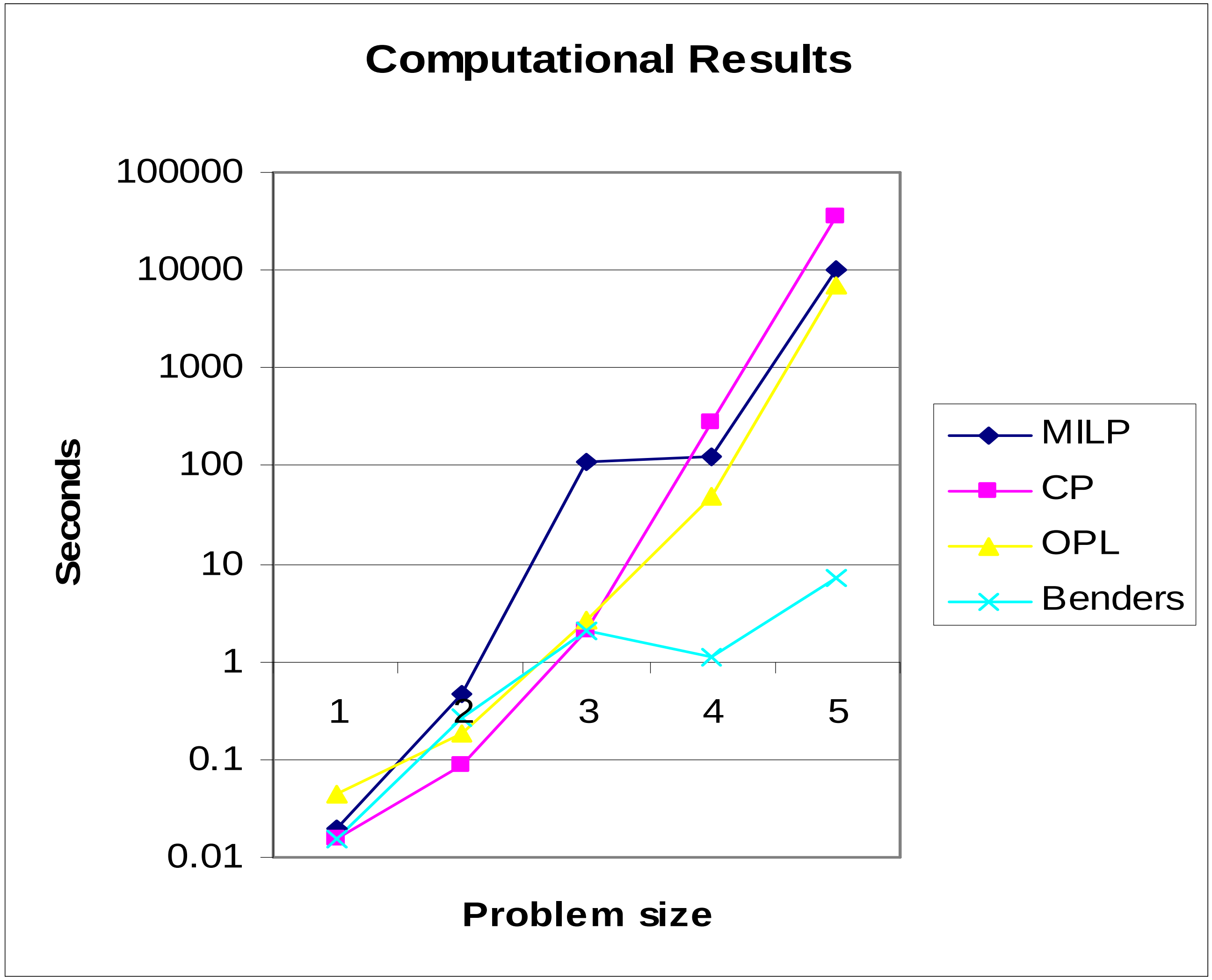
Valid constraint added to improve performance

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 has 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)*:

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

For $n = 4$:

$$x_1 + x_2 + x_3 + x_4 = 10$$
$$x_1 + x_2 + x_3 \ge 6, \quad x_1 + x_2 + x_4 \ge 6, \quad x_1 + x_3 + x_4 \ge 6, \quad x_2 + x_3 + x_4 \ge 6$$
$$x_1 + x_2 \ge 3, \quad x_1 + x_3 \ge 3, \quad x_1 + x_4 \ge 3, \quad x_2 + x_3 \ge 3, \quad x_2 + x_4 \ge 3, \quad x_3 + x_4 \ge 3$$
$$x_1, x_2, x_3, x_4 \ge 1$$

Relaxation of *element*

To implement variably indexed constant a_y

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

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):*

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

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

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

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

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

Visit each city exactly once in a single tour

y_j = city immediately following city j

Can also write:

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

y_j = jth city in tour

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

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

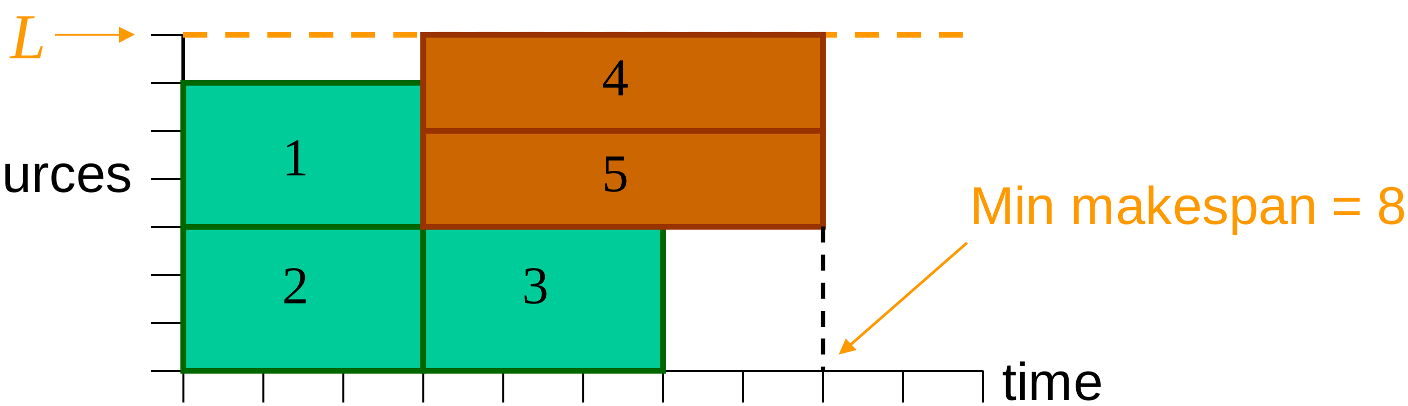
The following cut is valid for any subset of jobs $\{j_1,\ldots,j_k\}$

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

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:

$$
\begin{array}{ll}
\min & z \\
\text{s.t.} & z \geq t_1 + 3,\ t_2 + 3,\ t_3 + 3,\ t_4 + 5,\ t_5 + 5 \\
 & t_1 + t_2 + t_3 \geq 3 \\
 & t_1 + t_2 + t_3 + t_4 \geq 3\frac{5}{14} \\
 & t_2 + t_3 + t_4 + t_5 \geq 2\frac{4}{7} \\
 & t_1 + t_2 + t_3 + t_4 + t_5 \geq 6\frac{6}{7} \\
 & t_j \geq 0
\end{array}
$$

Facet defining

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)*.

$$\begin{aligned} & A^k x^k \le b^k y_k, \quad \text{all } k \\ & x = \sum_k x^k \\ & \sum_k y_k = 1 \\ & y_k \ge 0 \end{aligned}$$

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:

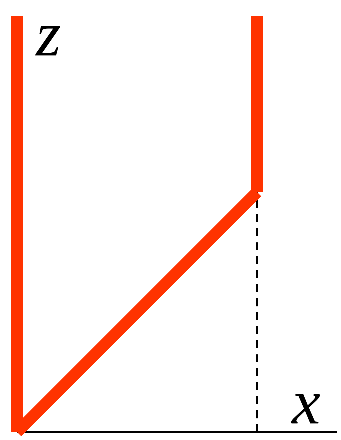
$$\begin{pmatrix} \text{no machine} \\ x = 0 \end{pmatrix} \vee \begin{pmatrix} \text{small machine} \\ z = 50 \\ x \le 5 \end{pmatrix} \vee \begin{pmatrix} \text{large machine} \\ z = 80 \\ x \le 10 \end{pmatrix}$$

Fixed cost of machine

Output of machine

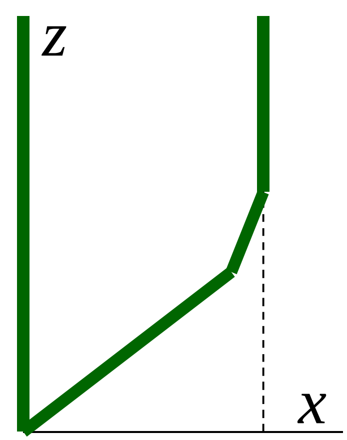
Convex hull relaxation:

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



Big-M relaxation:

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



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.)

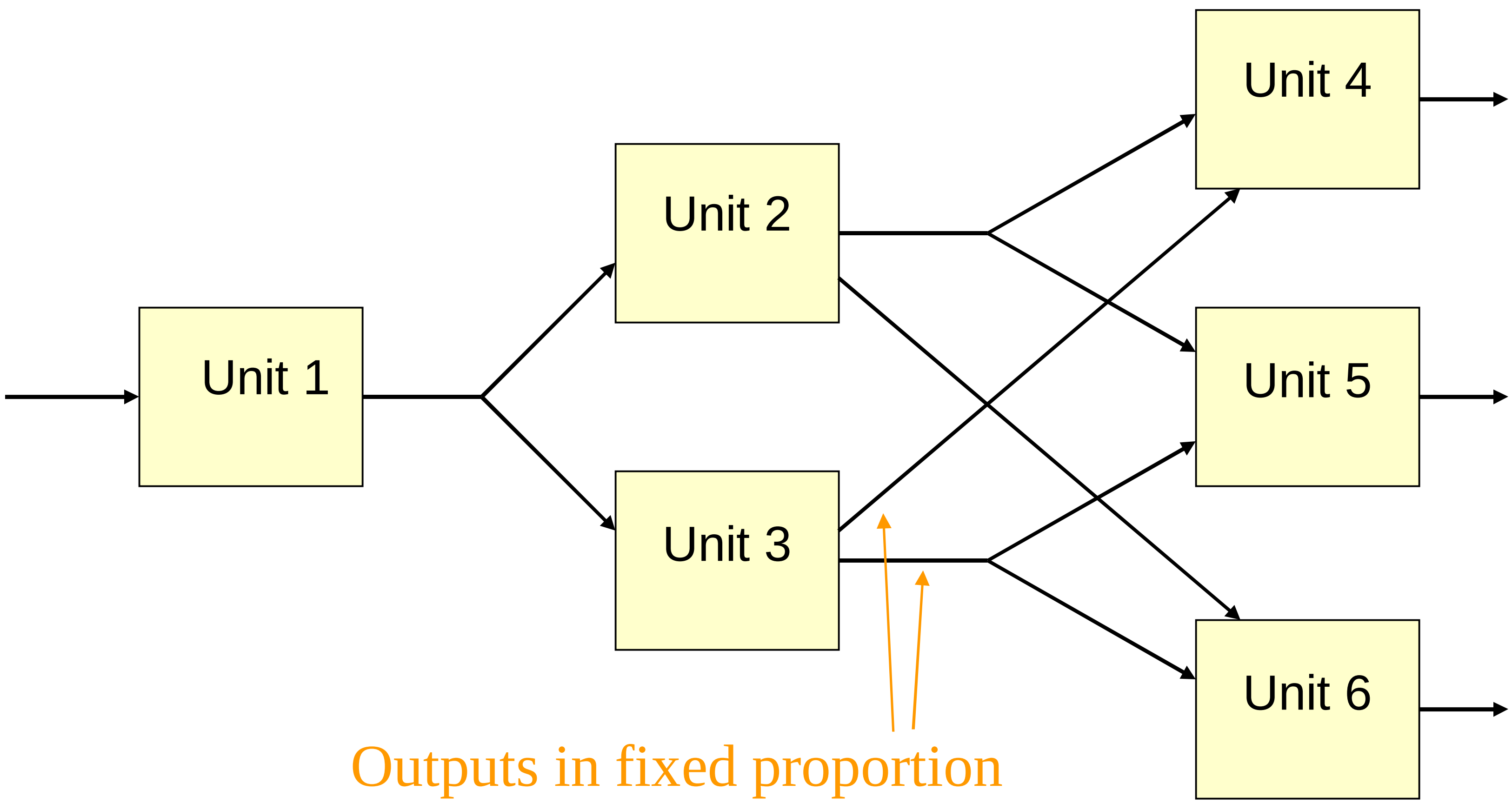
Elements of a General Scheme

- 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
$y_i \in D_i = \{\text{true},\text{false}\}$	presence or absence of unit i

Objective Function Window

$$\boxed{\max \sum_i (r_i u_i - z_i)}$$

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.**

$\boldsymbol{Ax} + \boldsymbol{Bu} = \boldsymbol{b}$ (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:

$$\begin{array}{ll} y_1 \rightarrow (y_2 \vee y_3) & y_3 \rightarrow y_4 \\ y_2 \rightarrow y_1 & y_3 \rightarrow (y_5 \vee y_6) \\ y_2 \rightarrow (y_4 \vee y_5) & y_4 \rightarrow (y_2 \vee y_3) \\ y_2 \rightarrow y_6 & y_5 \rightarrow (y_2 \vee y_3) \\ y_3 \rightarrow y_1 & y_6 \rightarrow (y_2 \vee y_3) \end{array}$$

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.

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*, most recent issue.
- 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.