Integrating Constraint Programming and Mathematical Programming

John Hooker

Carnegie Mellon University

University of York, June 2003

These slides are available at

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

g

A Simple Example

Integrating CP and MP

Branch Infer and Relax

Decomposition

Recent Success Stories

Relaxation

Putting it Together

Surveys and Tutorials

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.

A Simple Example

Solution by Constraint Programming
Solution by Integer Programming
Solution by a Hybrid Method

The Problem

$$\begin{array}{ll} \min & 5x_1 + 8x_2 + 4x_3 \\ \text{subject to} & 3x_1 + 5x_2 + 2x_3 \geq 30 \\ & \text{all-different}\{x_1, x_2, x_3\} \\ & x_j \in \{1, \ldots, 4\} \end{array}$$

We will illustrate how search, inference and relaxation 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
Relaxation: Constraint store (set of current variable domains)

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

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

Global constraint

Constraint store can be viewed as consisting of in-domain constraints $x_j \in D_j$, which form a relaxation of the problem.

Domain reduction for inequalities

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

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

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

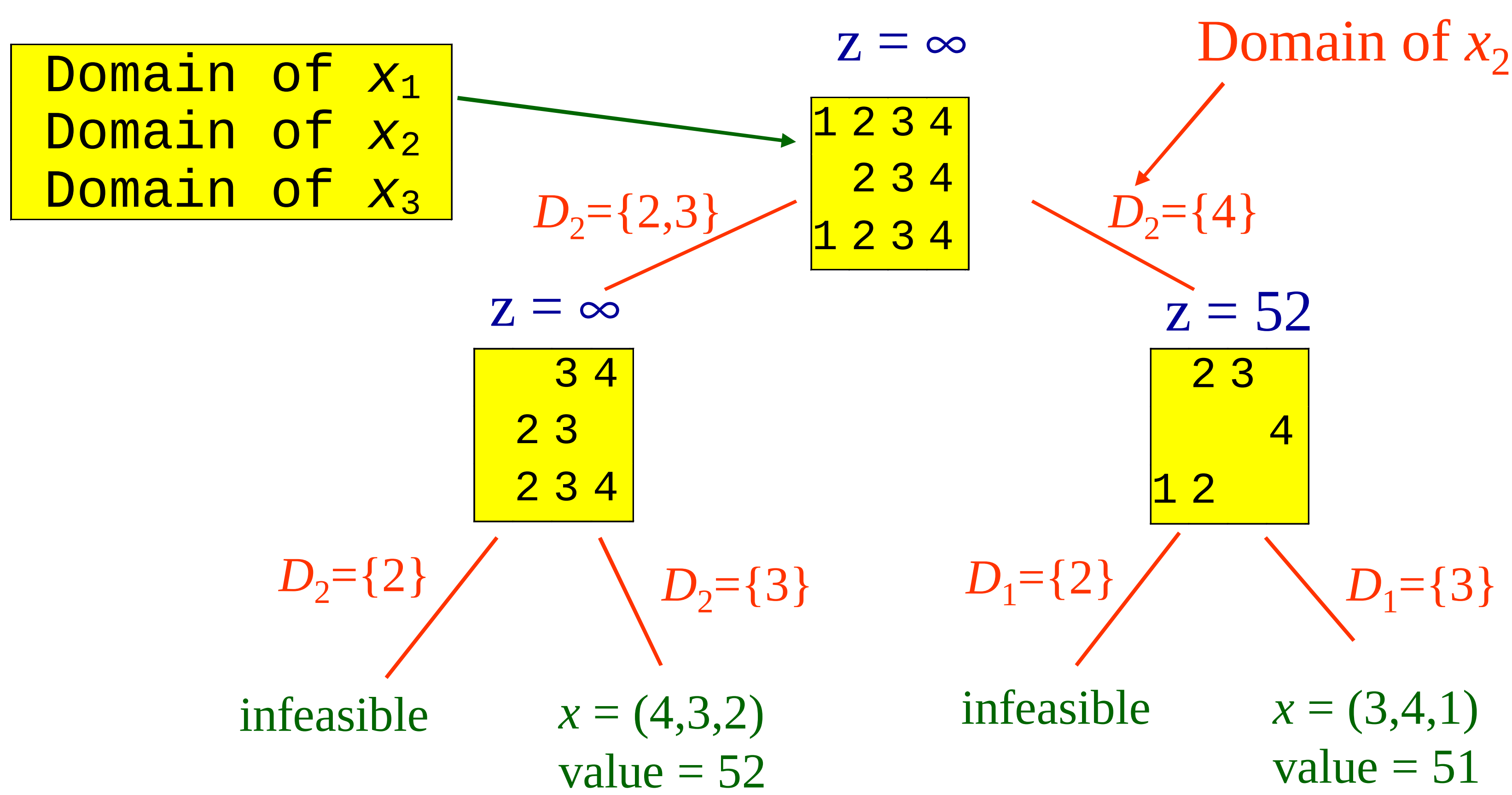
Suppose for example:

```
1 2        ← Domain of x1
1 2        ← Domain of x2
1 2 3 4    ← Domain of x3
```

Then one can reduce the domains:

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

- In general, solve a maximum cardinality matching problem and apply a theorem of Berge



Domain of x_1
Domain of x_2
Domain of x_3
z = ∞
1 2 3 4
2 3 4
1 2 3 4
Domain of x_2
D_2={2,3}
D_2={4}
z = ∞
3 4
2 3
2 3 4
z = 52
2 3
4
1 2
D_2={2}
D_2={3}
D_1={2}
D_1={3}
infeasible
x = (4,3,2)
value = 52
infeasible
x = (3,4,1)
value = 51

Solve as an integer programming problem

Search: Branch on variables with fractional values in solution of continuous relaxation.

Inference: Generate cutting planes (covering inequalities).

Relaxation: Continuous (LP) relaxation.

Rewrite problem using integer programming model:

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

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

Continuous relaxation

$$\begin{array}{ll}
\min & 4x_1 + 3x_2 + 5x_3 \\
\text{subject to} & 4x_1 + 2x_2 + 4x_3 \geq 17 \\
& x_i = \sum_{j=1}^{4} j y_{ij}, \quad i = 1,2,3 \\
& \sum_{j=1}^{4} y_{ij} = 1, \quad i = 1,2,3 \\
& \sum_{i=1}^{3} y_{ij} \leq 1, \quad j = 1,\ldots,4 \\
& x_1 + x_2 \geq 5 \\
\text{Covering inequalities} & x_1 + x_3 \geq 4 \\
& x_2 + x_3 \geq 4 \\
& x_1 + x_2 + x_3 \geq 8 \\
\text{Relax integrality} & 0 \leq y_{ij} \leq 1, \quad \text{all } i, j
\end{array}$$

Branch and bound (Branch and relax)

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

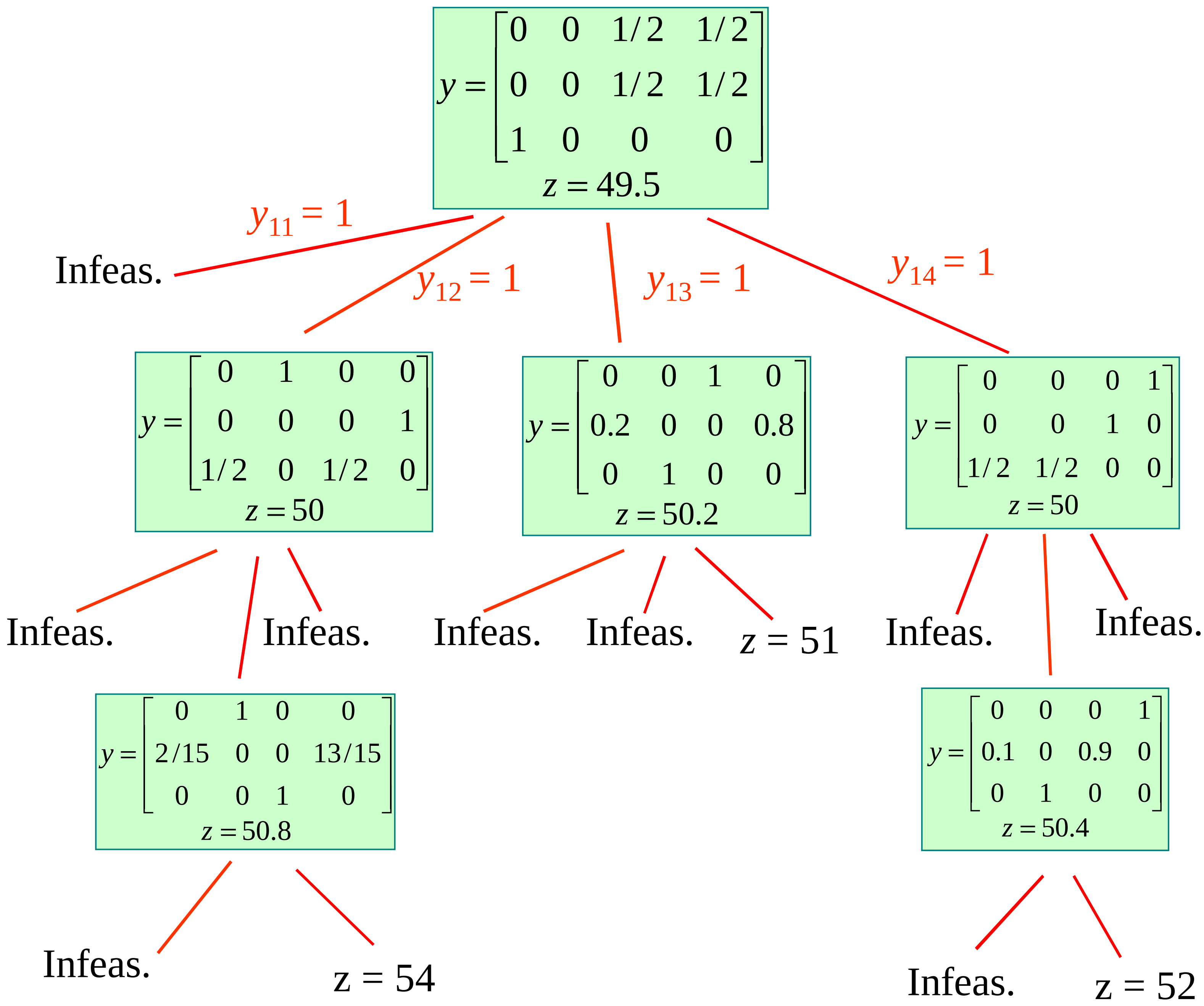
At each node of the branching tree:

- 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.

- Use SOS-1 branching.



y = [0 0 1/2 1/2; 0 0 1/2 1/2; 1 0 0 0]
z = 49.5
y11 = 1
y12 = 1
y13 = 1
y14 = 1
Infeas.
y = [0 1 0 0; 0 0 0 1; 1/2 0 1/2 0]
z = 50
y = [0 0 1 0; 0.2 0 0 0.8; 0 1 0 0]
z = 50.2
y = [0 0 0 1; 0 0 1 0; 1/2 1/2 0 0]
z = 50
Infeas.
Infeas.
Infeas.
Infeas.
z = 51
Infeas.
Infeas.
y = [0 1 0 0; 2/15 0 0 13/15; 0 0 1 0]
z = 50.8
y = [0 0 0 1; 0.1 0 0.9 0; 0 1 0 0]
z = 50.4
Infeas.
z = 54
Infeas.
z = 52

Solve using a hybrid approach

Search:

- Branch on fractional variables in solution of relaxation.
  - Drop constraints 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 for all inequalities.
- Maintain hyperarc consistency for all-different constraints.

Relaxation:

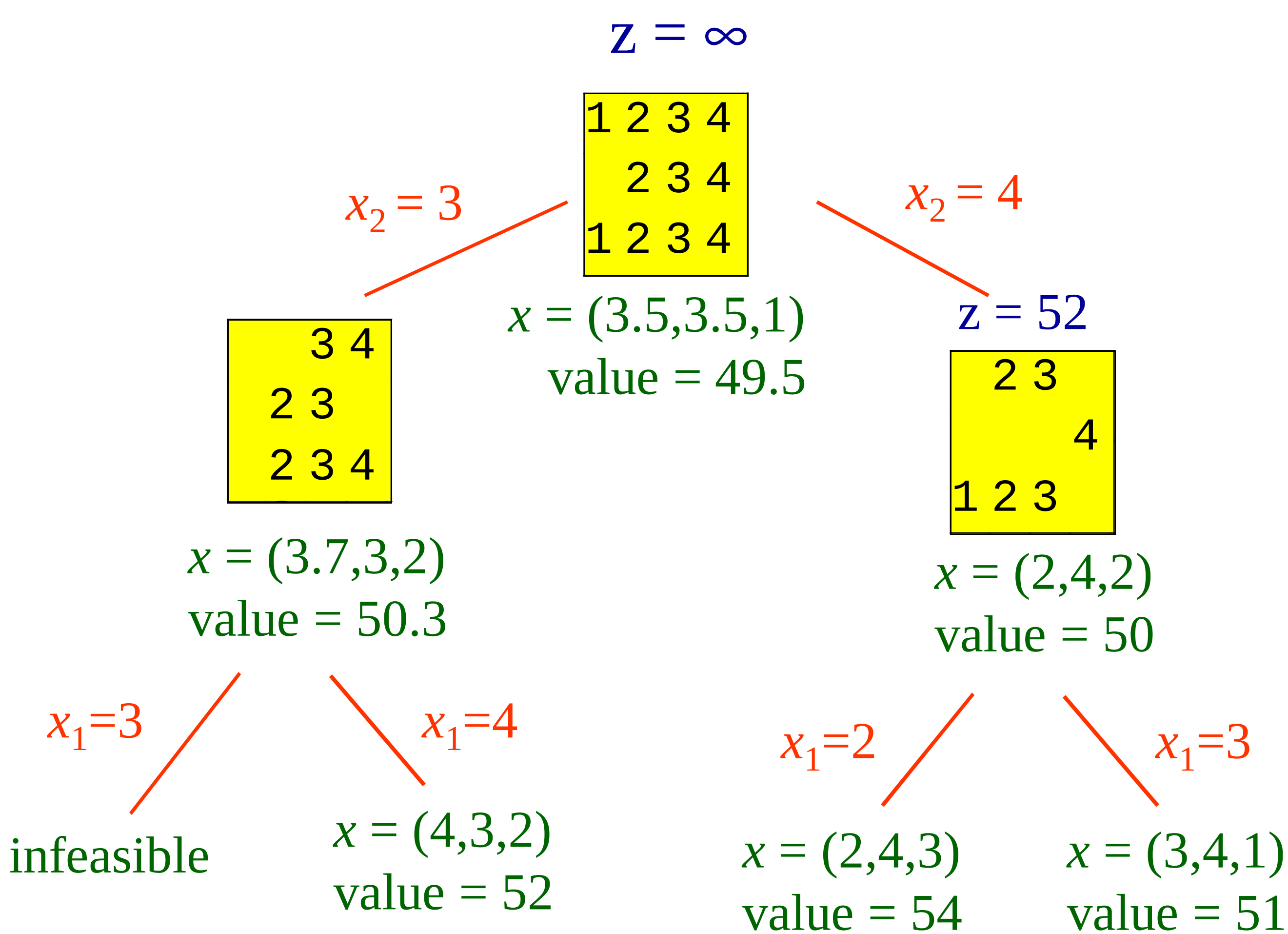
- Put knapsack constraint in LP.
- Put covering inequalities based on knapsack/all-different into LP.

Model for hybrid approach

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

Covering inequalities

Generate and propagate covering inequalities at each node of search tree


z = ∞
1 2 3 4
2 3 4
1 2 3 4
x = (3.5,3.5,1)
value = 49.5
x2 = 3
x2 = 4
3 4
2 3
2 3 4
x = (3.7,3,2)
value = 50.3
z = 52
2 3
4
1 2 3
x = (2,4,2)
value = 50
x1=3
infeasible
x1=4
x = (4,3,2)
value = 52
x1=2
x = (2,4,3)
value = 54
x1=3
x = (3,4,1)
value = 51

Integrating CP and MP

Motivation
Two Integration Schemes

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 two integrative ideas:

- *Branch-infer-and-relax* - View CP and MP methods as special cases of a branch-infer-and-relax method.
- *Decomposition* - Decompose problems into a CP part and an MP part, perhaps using a Benders scheme.

Branch-infer-and-relax

- Existing CP and MP combine branching, inference and relaxation.
  - *Branching* – enumerate solutions by branching on variables or violated constraints.
  - *Inference* – deduce new constraints
    - CP: domain reduction
    - MP: cutting planes.
  - *Relaxation* – remove some constraints before solving
    - CP: the constraint store (variable domains)
    - MP: continuous relaxation

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.

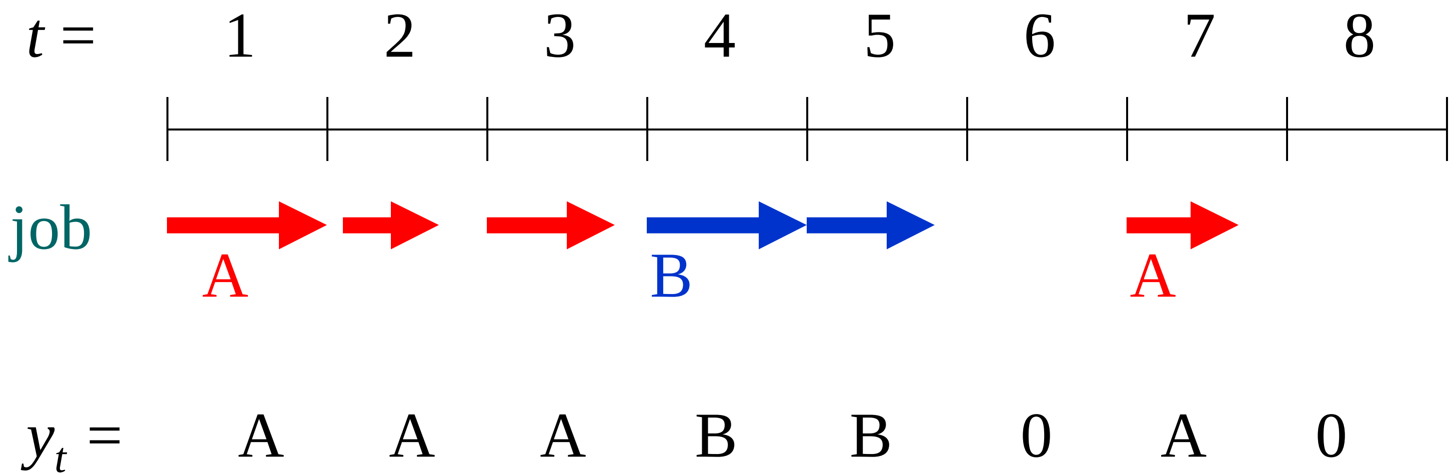
Branch Infer and Relax

A Second Example: Discrete Lot Sizing

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



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

inventory balance

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

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

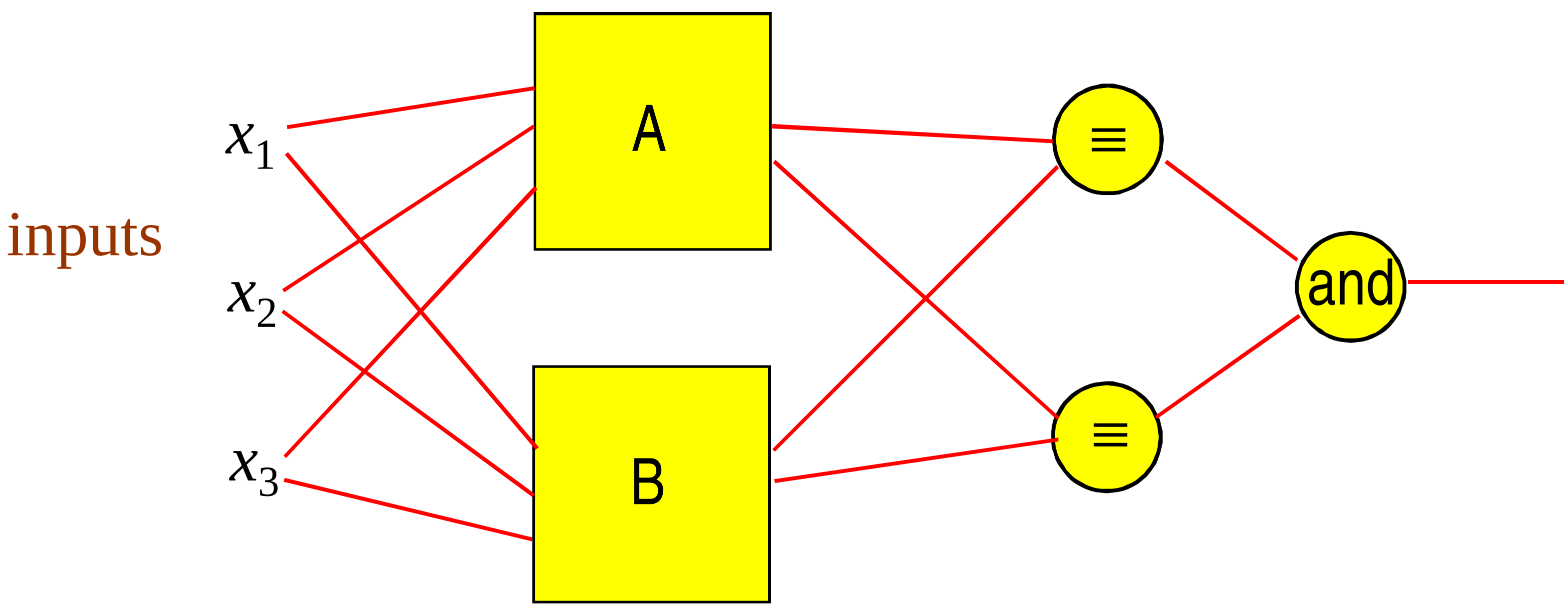
“Learn from one’s mistakes.”

- 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

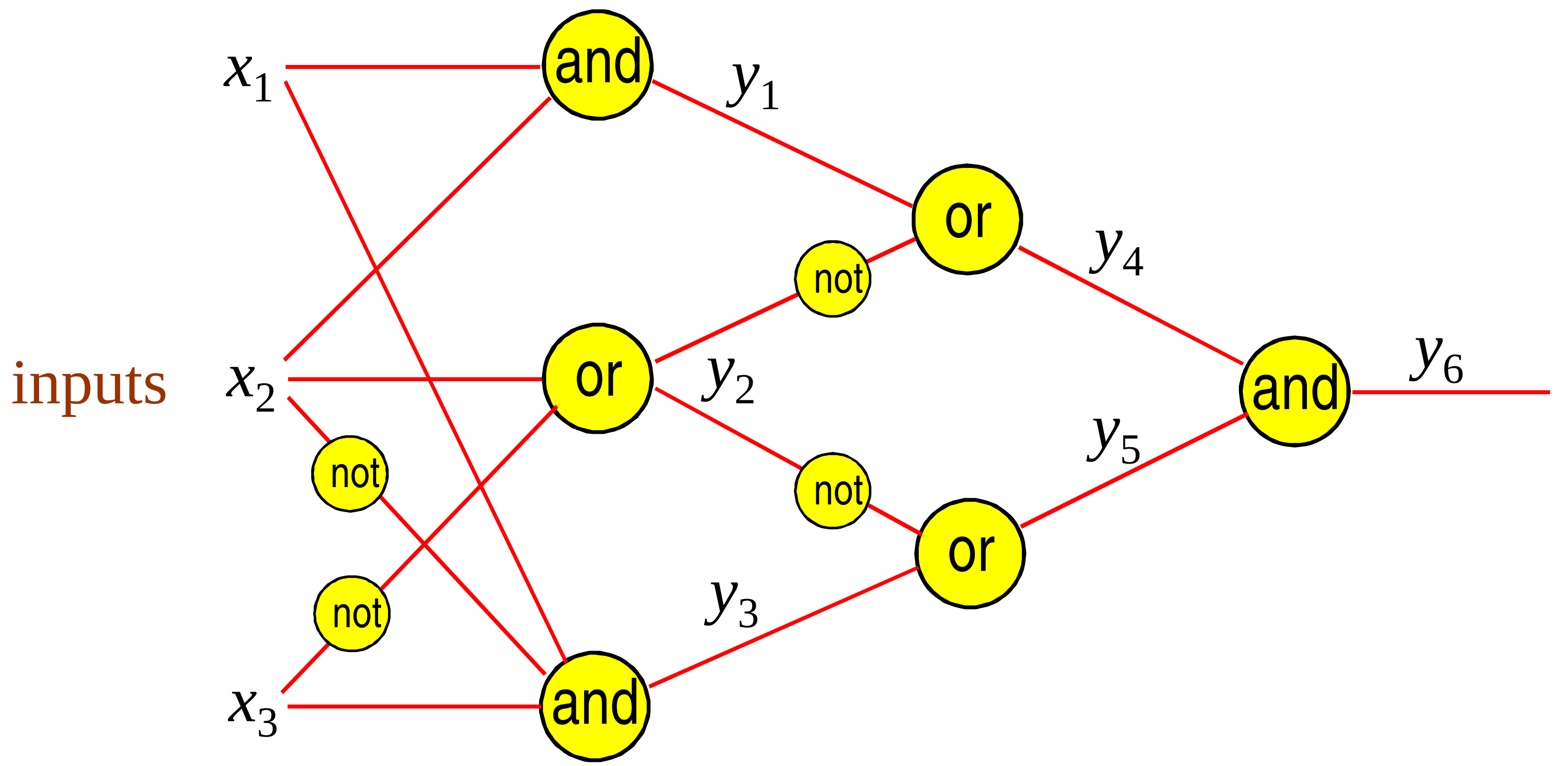
(JNH, Yan)

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.

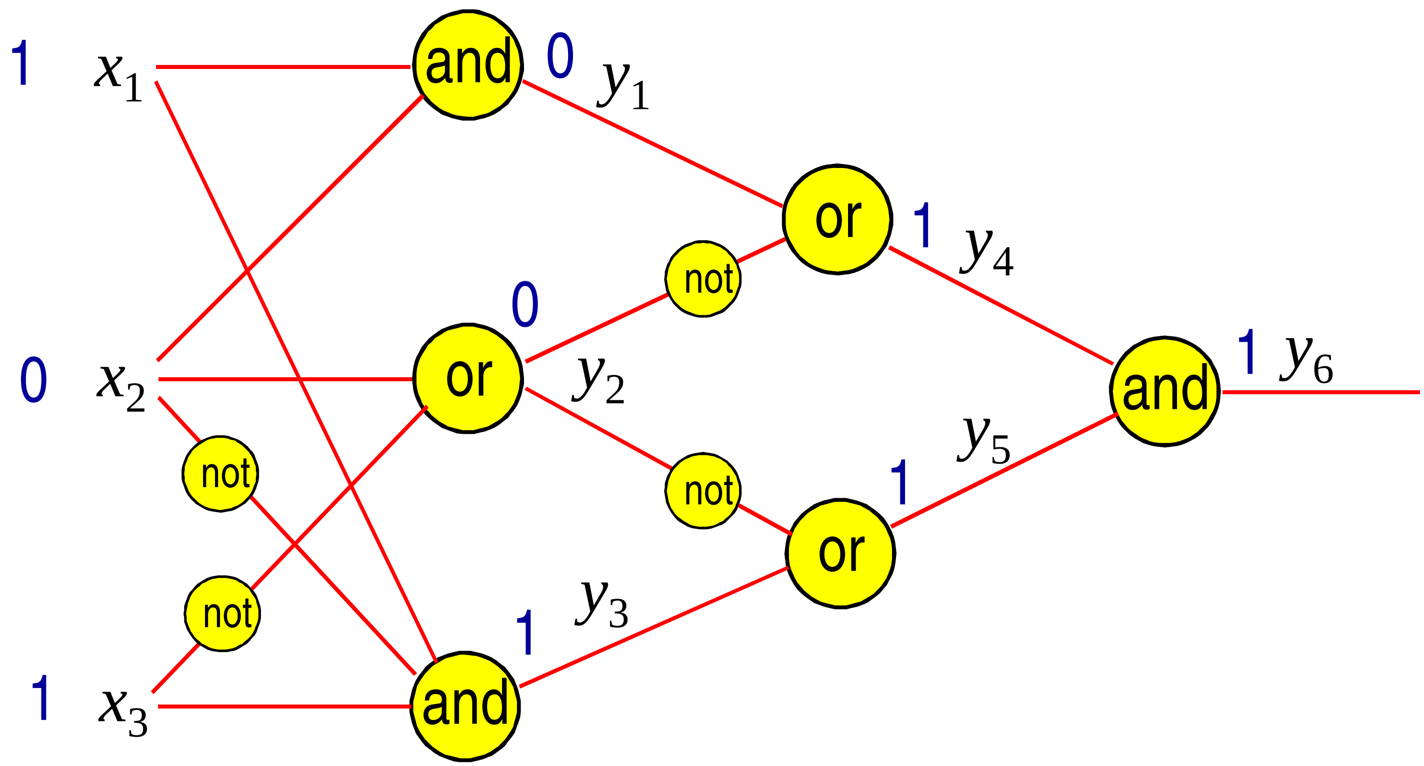
For instance, check whether this circuit is a tautology:



The subproblem is to find whether the output can be 0 when the input x is fixed to a given value.

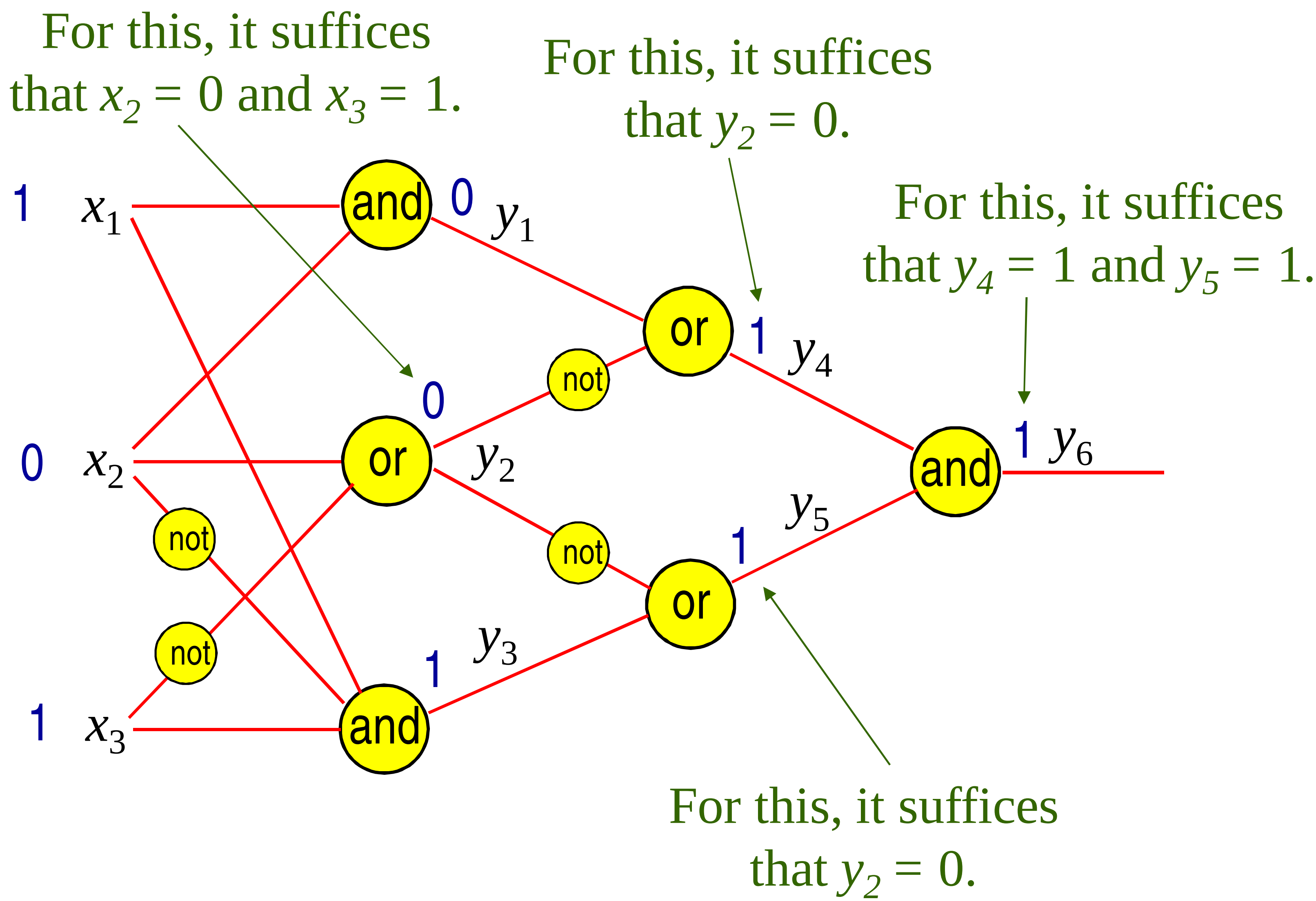
But since x determines the output of the circuit, the subproblem is easy: just compute the output.

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.



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.

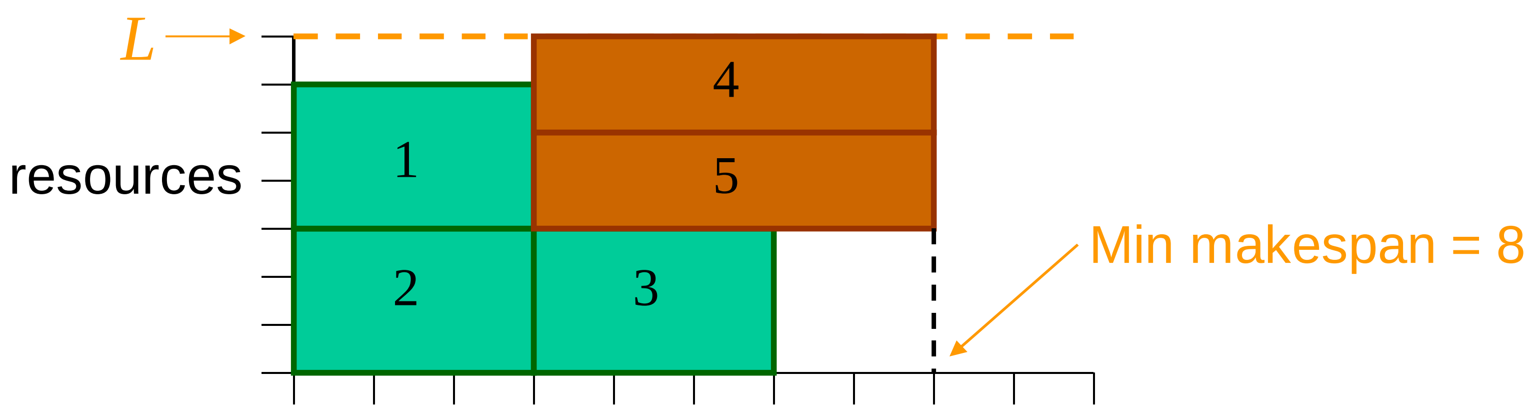
- In this problem, the master problem assigns jobs to machines. The subproblem schedules jobs assigned to each machine.
- Classical mixed integer programming solves the master problem.
- Constraint programming solves the subproblem, a 1-machine scheduling problem with time windows.
- This provides a 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 Durations Resource consumption 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,

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

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

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

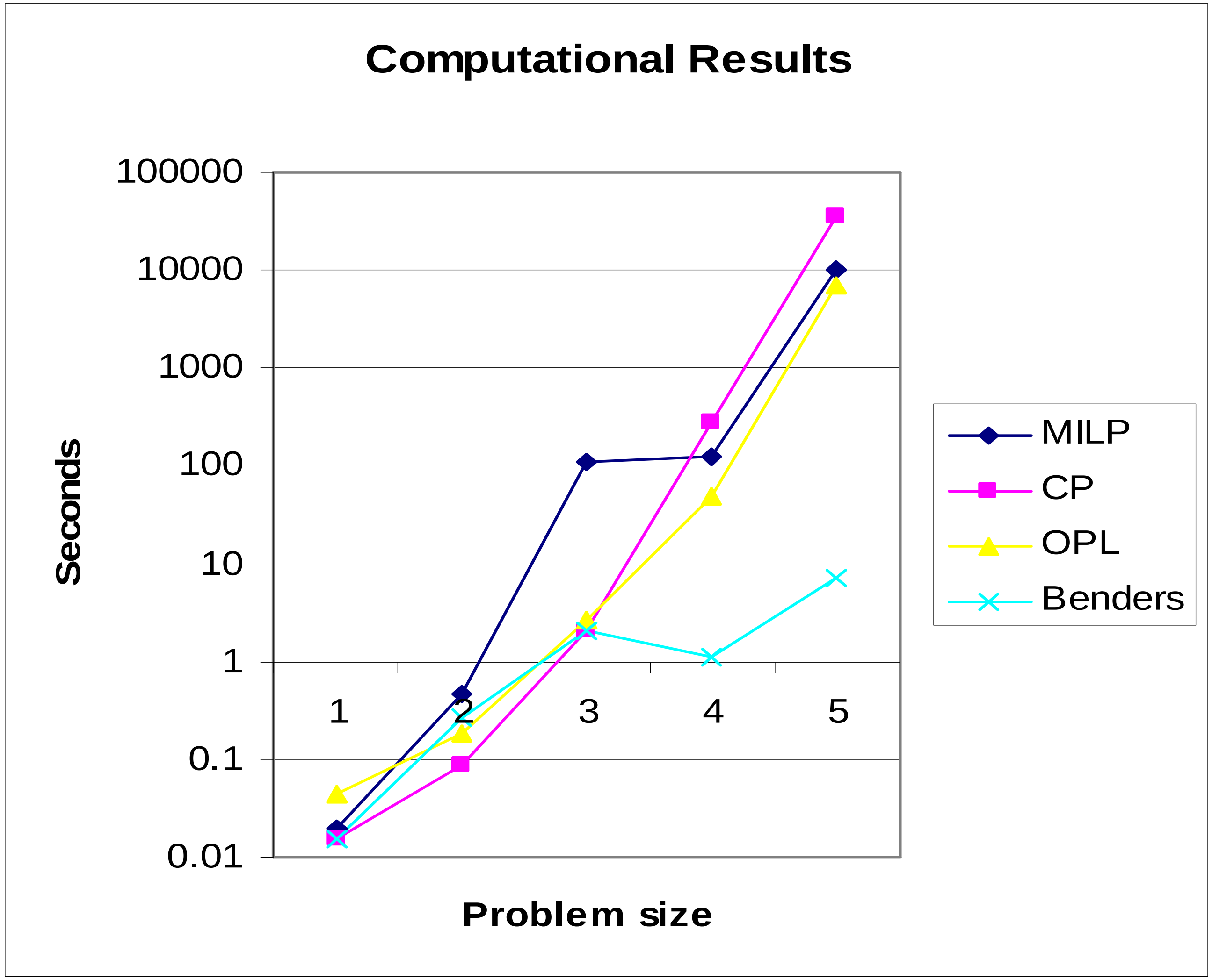
Valid constraint added to improve performance

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

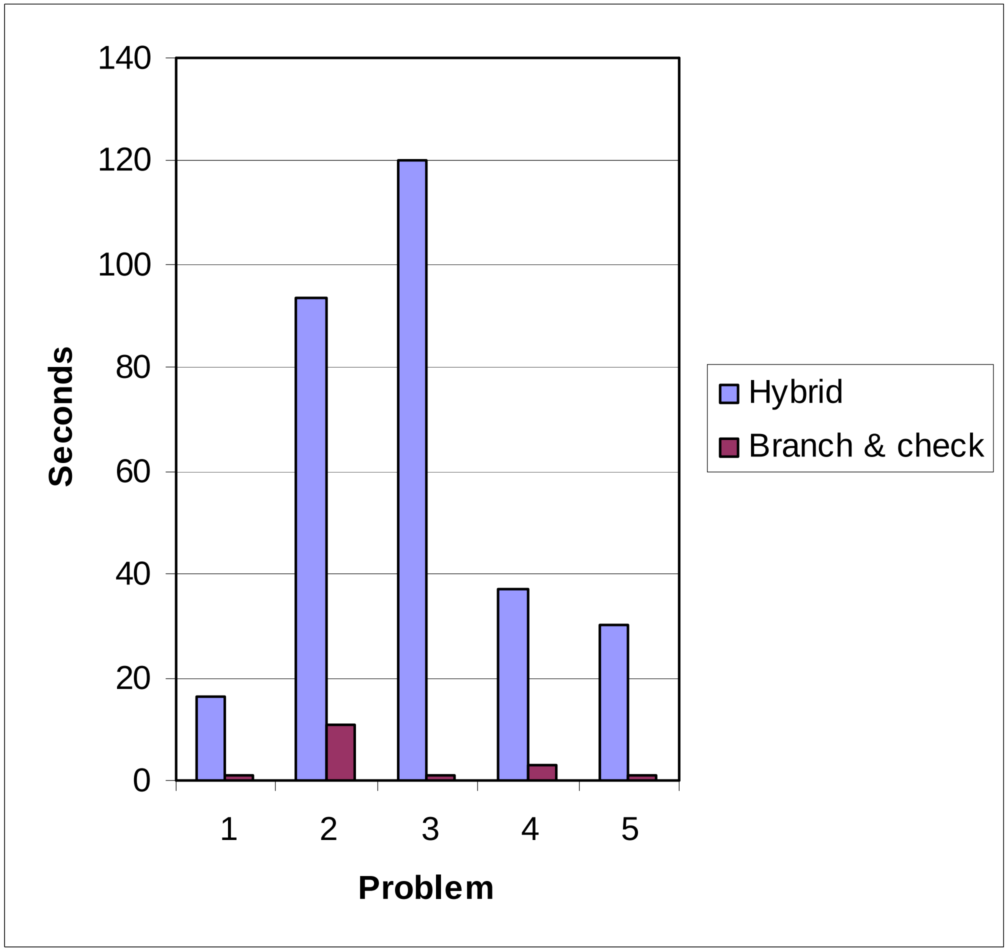


An Enhancement: Branch and Check
(JNH, Thorsteinsson)

- 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.

Enhancement Using "Branch and Check" *(Thorsteinsson)*

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



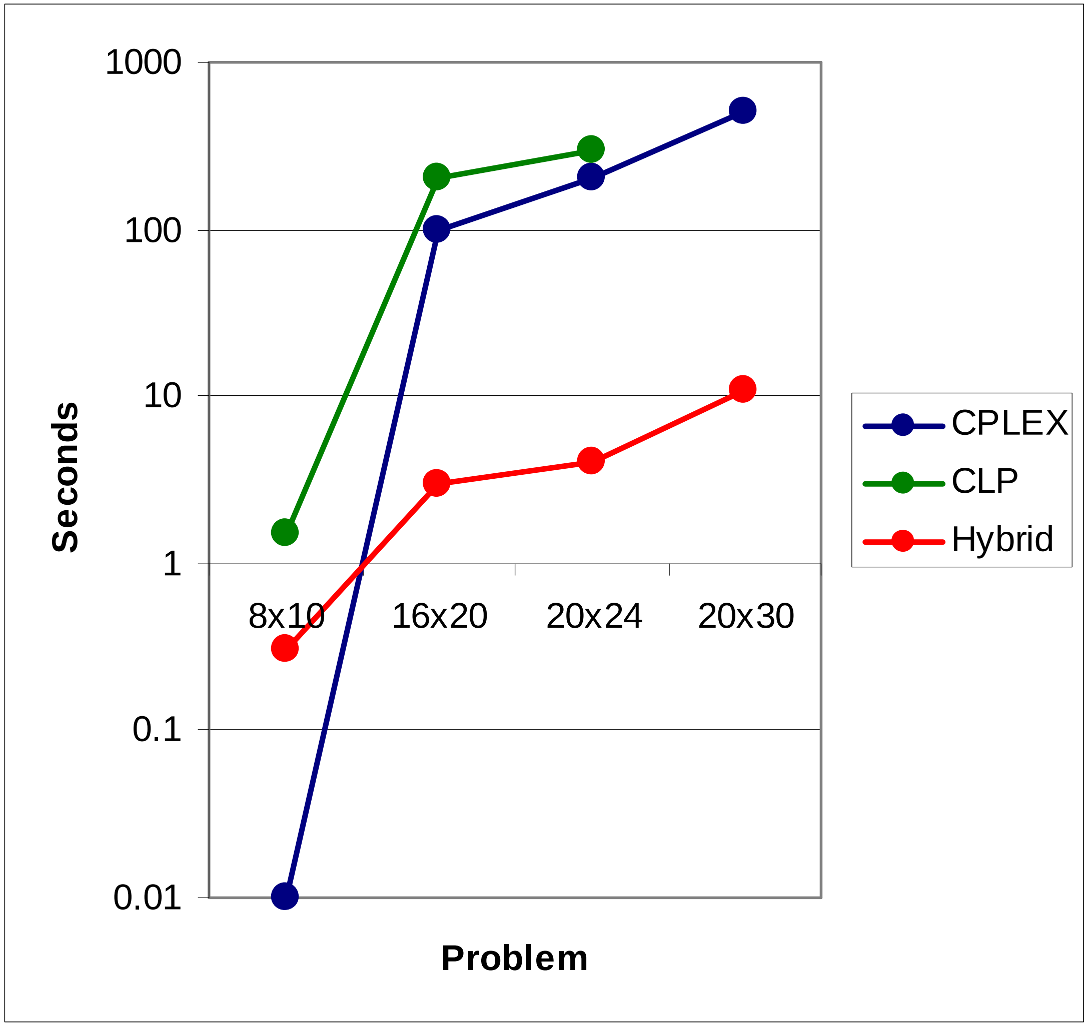
Recent Success Stories

Product Configuration
Process Scheduling at BASF
Paint Production at Barbot
Production Line Sequencing at Peugeot/Citroën
Line Balancing at Peugeot/Citroën

Example of Faster Solution: Product configuration

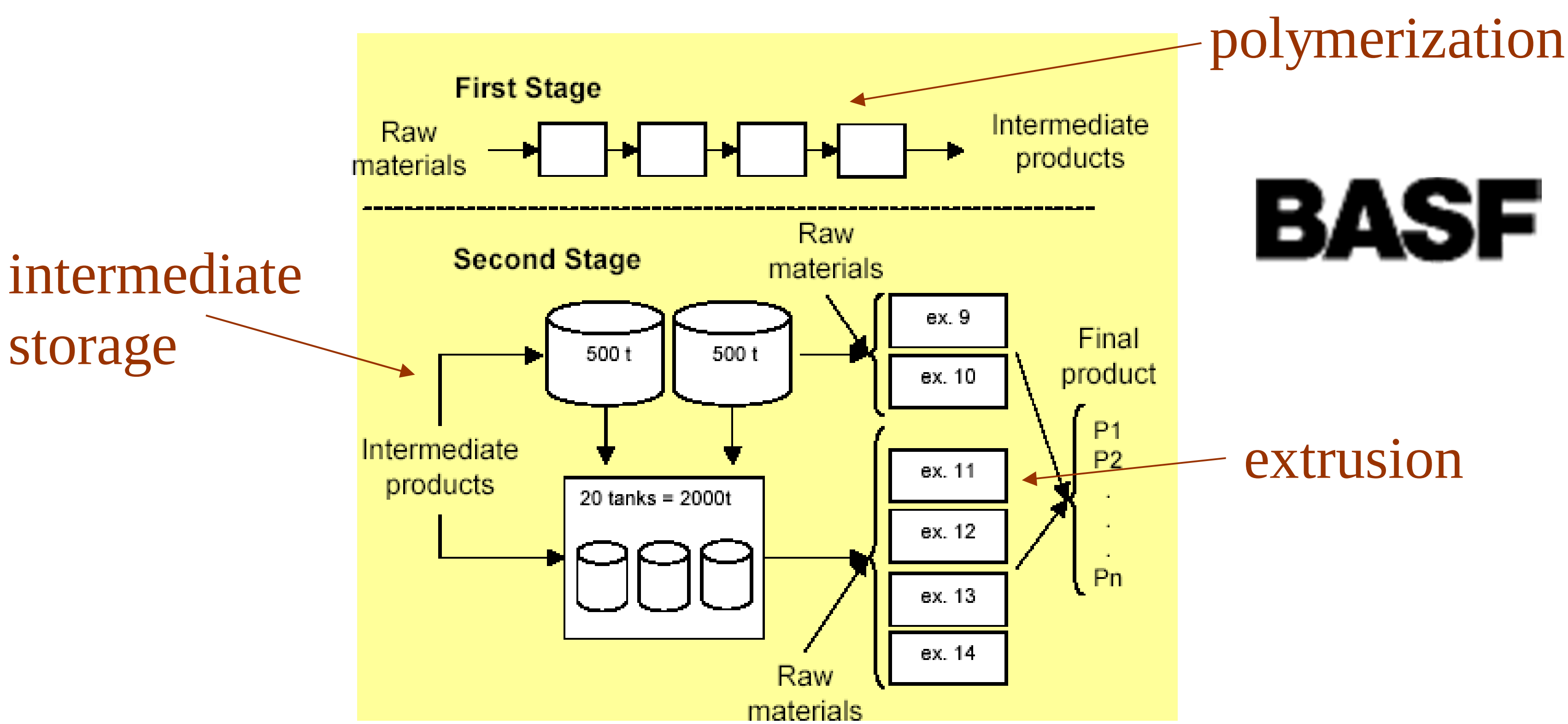
- Find optimal selection of components to make up a product, subject to configuration constraints.
- Use continuous relaxation of element constraints and reduced cost propagation.

Computational Results *(Ottosson & Thorsteinsson)*



Process Scheduling and Lot Sizing at BASF

Manufacture of polypropylenes in 3 stages



Process Scheduling and Lot Sizing at BASF

- Manual planning (old method)
  - Required 3 days
  - Limited flexibility and quality control

- 24/7 continuous production
  - Variable batch size.
  - Sequence-dependent changeover times.

Process Scheduling and Lot Sizing at BASF

- Intermediate storage
  - Limited capacity
  - One product per silo
- Extrusion
  - Production rate depends on product and machine

Process Scheduling and Lot Sizing at BASF

- Three problems in one
  - Lot sizing – based on customer demand forecasts
  - Assignment – put each batch on a particular machine
  - Sequencing – decide the order in which each machine processes batches assigned to it

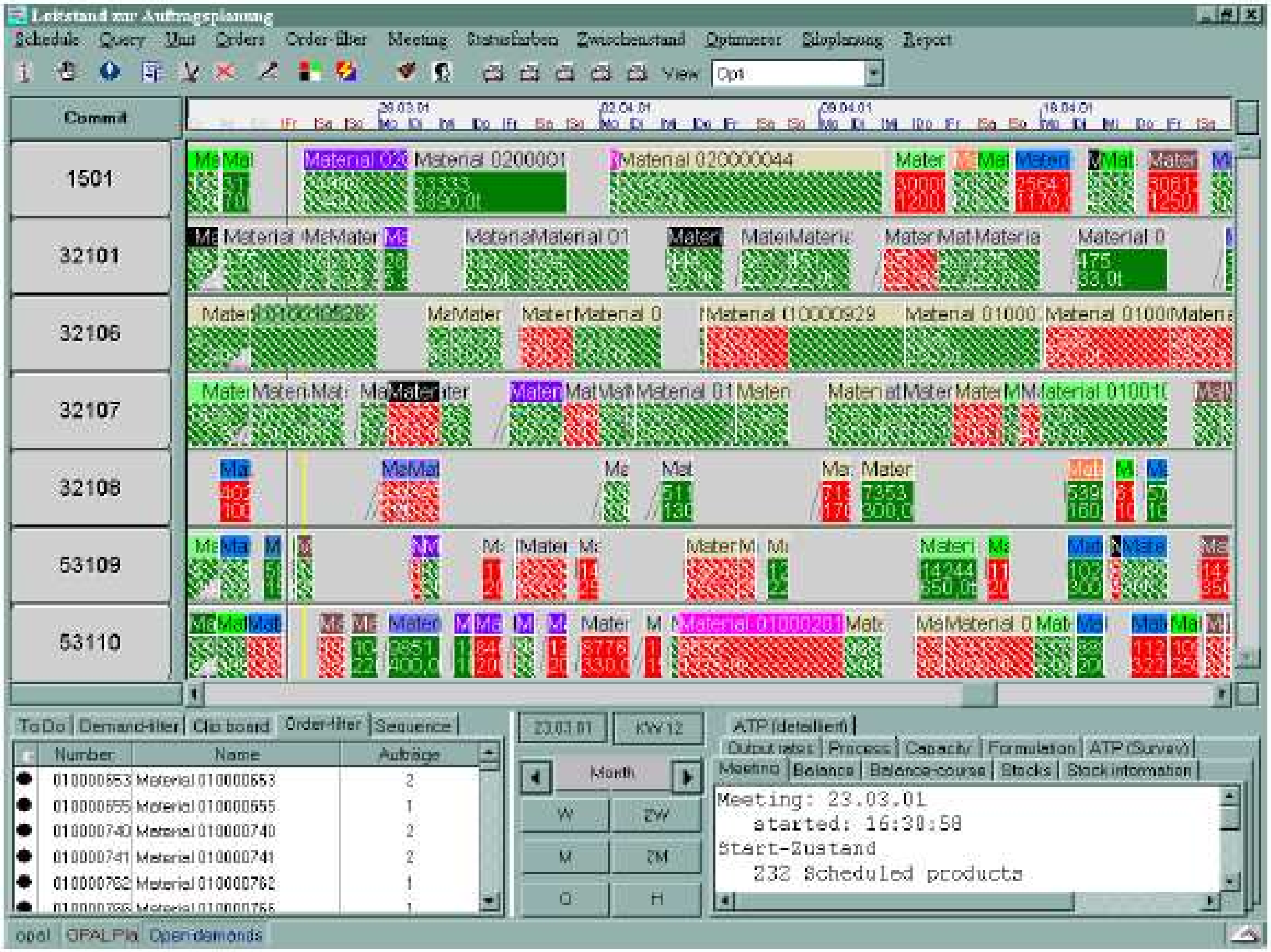
Process Scheduling and Lot Sizing at BASF

- The problems are interdependent
  - Lot sizing depends on assignment, since machines run at different speeds
  - Assignment depends on sequencing, due to restrictions on changeovers
  - Sequencing depends on lot sizing, due to limited intermediate storage

Process Scheduling and Lot Sizing at BASF

- Solve the problems simultaneously
  - *Lot sizing:* solve with MIP (using XPRESS-MP)
  - *Assignment:* solve with MIP
  - *Sequencing:* solve with CP (using CHIP)
- The MIP and CP are linked mathematically.
  - Use logic-based Benders decomposition, developed only in the last few years.

Sample schedule, illustrated with Visual Scheduler (AviS/3)

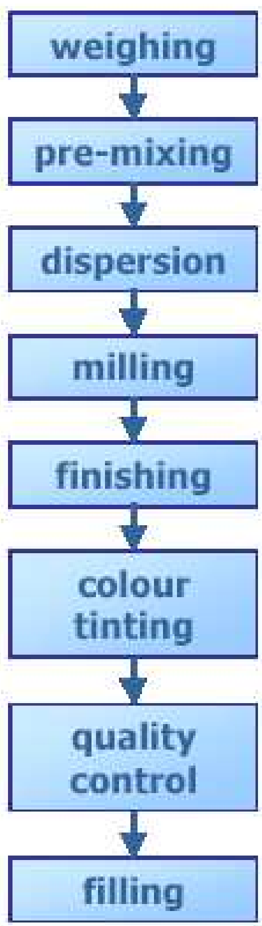


Source: BASF

Process Scheduling and Lot Sizing at BASF

- Benefits
  - Optimal solution obtained in 10 mins.
  - Entire planning process (data gathering, etc.) requires a few hours.
  - More flexibility
  - Faster response to customers
  - Better quality control

Paint Production at Barbot



Several machines of each type

- Two problems to solve simultaneously
  - Lot sizing
  - Machine scheduling
- Focus on solvent-based paints, for which there are fewer stages.
- Barbot is a Portuguese paint manufacturer.

TINTAS BARBOT INDÚSTRIA DE TINTAS S.A.

Paint Production at Barbot

- Solution method similar to BASF case (MIP + CP).
- Benefits
  - Optimal solution obtained in a few minutes for 20 machines and 80 products.
  - Product shortages eliminated.
  - 10% increase in output.
  - Fewer cleanup materials.
  - Customer lead time reduced.

Production Line Sequencing at Peugeot/Citroën

- The Peugeot 206 can be manufactured with 12,000 option combinations.
- Planning horizon is 5 days

PSA PEUGEOT CITROËN

Production Line Sequencing at Peugeot/Citroën

- Each car passes through 3 shops.

- Objectives
  - Group similar cars (e.g. in paint shop).
  - Reduce setups.
  - Balance work station loads.

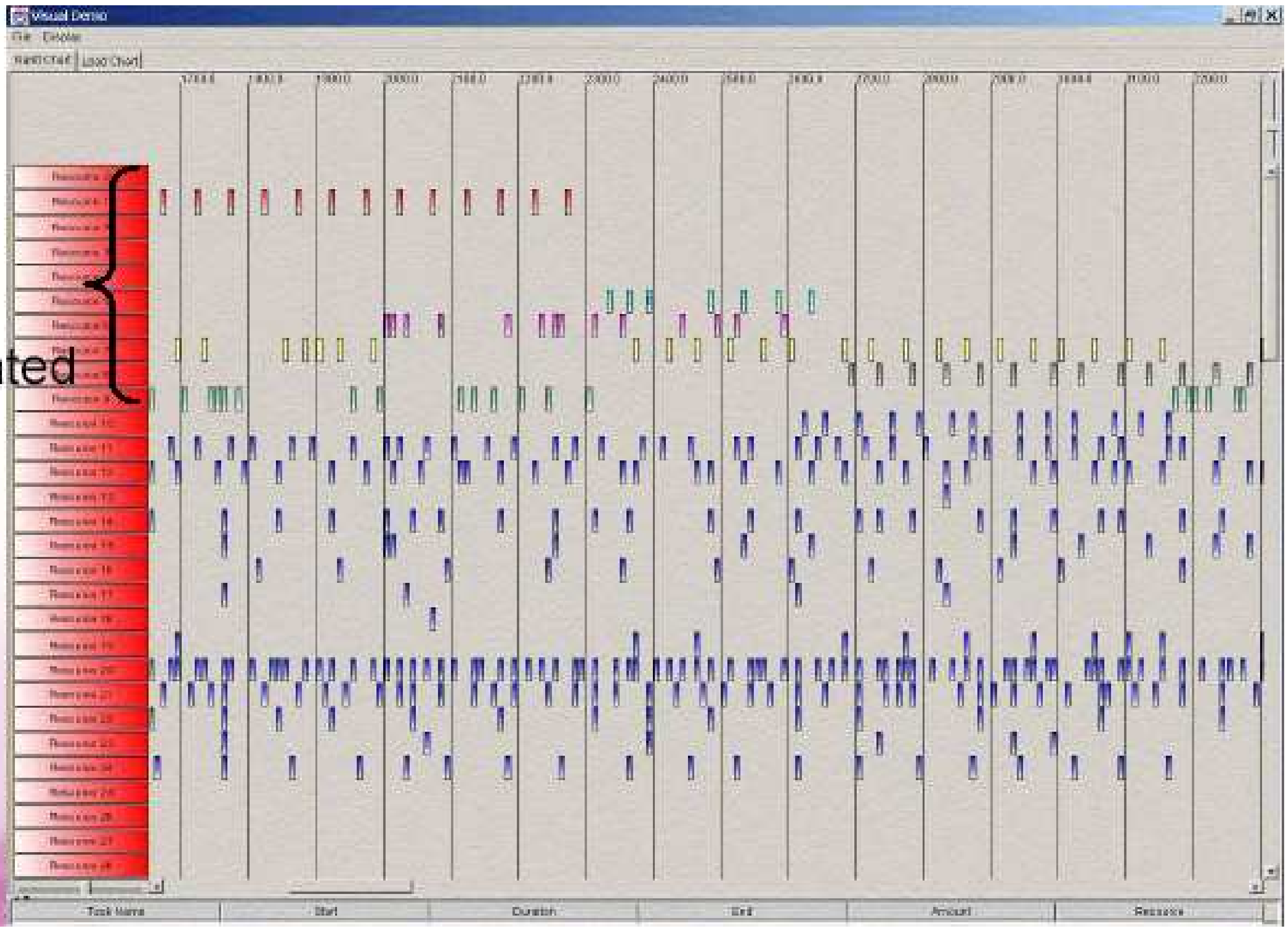
Production Line Sequencing at Peugeot/Citroën

- Special constraints
  - Cars with a sun roof should be grouped together in assembly.
  - Air-conditioned cars should not be assembled consecutively.
  - Etc.

Production Line Sequencing at Peugeot/Citroën

- Problem has two parts
  - Determine number of cars of each type assigned to each line on each day.
  - Determine sequencing for each line on each day.
- Problems are solved simultaneously.
  - Again by MIP + CP.

Sample schedule



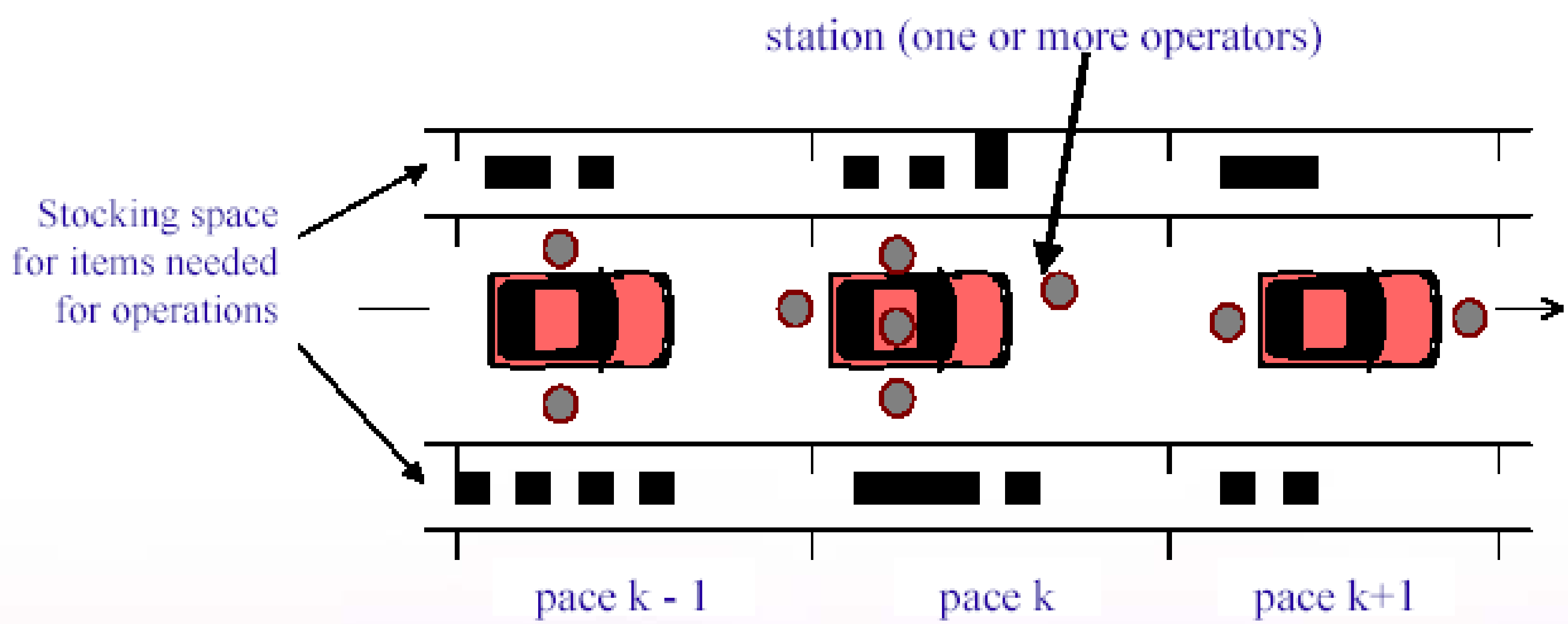
Source: Peugeot/Citroën

Production Line Sequencing at Peugeot/Citroën

- Benefits
  - Greater ability to balance such incompatible benefits as fewer setups and faster customer service.
  - Better schedules.

Line Balancing at Peugeot/Citroën

A classic production sequencing problem



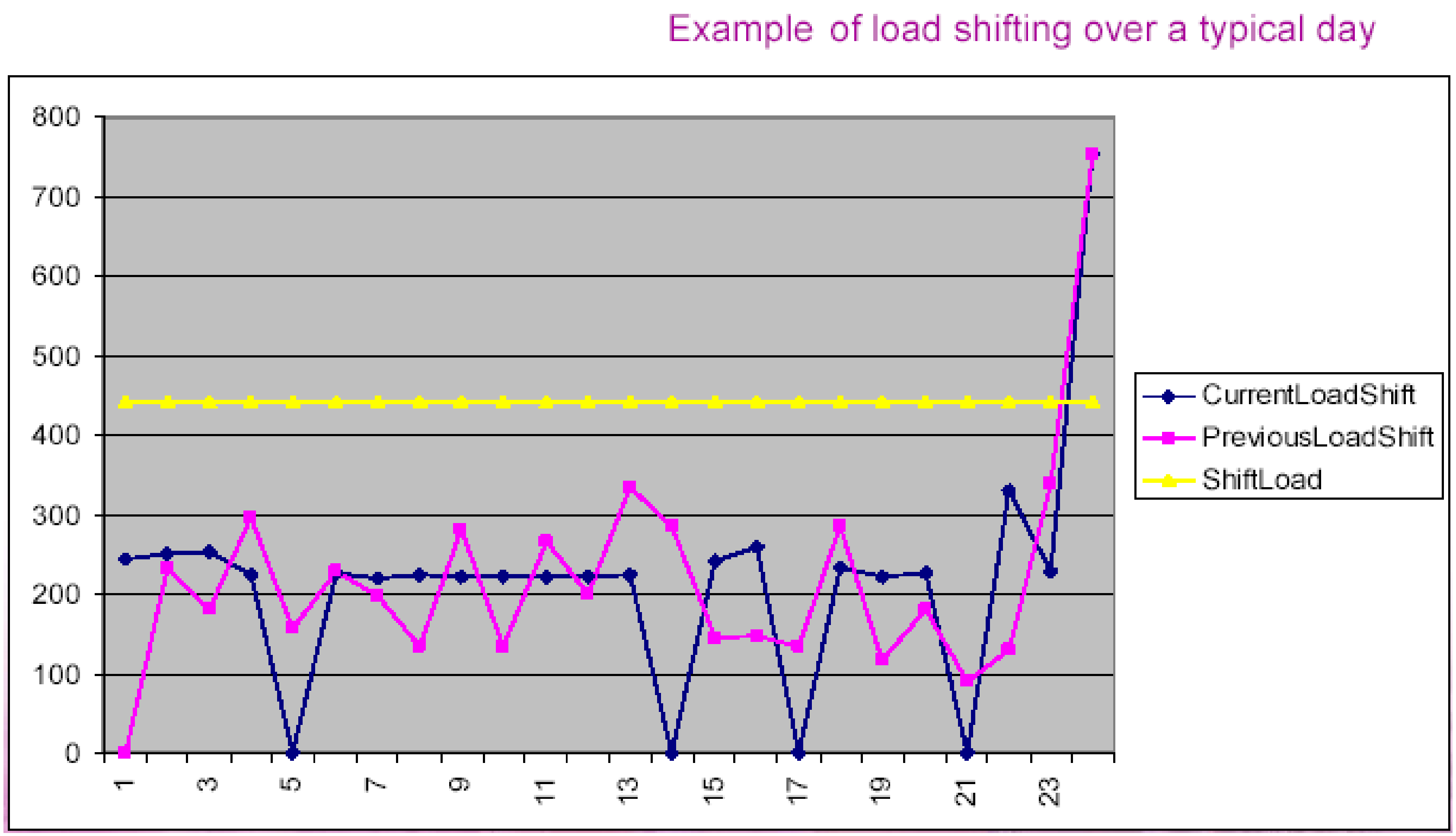
Source: Peugeot/Citroën

Line Balancing at Peugeot/Citroën

- Objective
  - Equalize load at work stations.
  - Keep each worker on one side of the car
- Constraints
  - Precedence constraints between some operations.
  - Ergonomic requirements.
  - Right equipment at stations (e.g. air socket)

Line Balancing at Peugeot/Citroën

- Solution again obtained by a hybrid method.
  - MIP: obtain solution without regard to precedence constraints.
  - CP: Reschedule to enforce precedence constraints.
  - The two methods interact.



Source: Peugeot/Citroën

Line Balancing at Peugeot/Citroën

- Benefits
  - Better equalization of load.
  - Some stations could be closed, reducing labor.
- Improvements needed
  - Reduce trackside clutter.
  - Equalize space requirements.
  - Keep workers on one side of car.

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)*:

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

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 $\text{element}\big(y, (a_1, \ldots, a_n), z\big)$

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 $\text{element}\big(y,(x_1,\ldots,x_n),z\big)$
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):*

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

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

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

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

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 $\{j_1,\ldots,j_k\}$

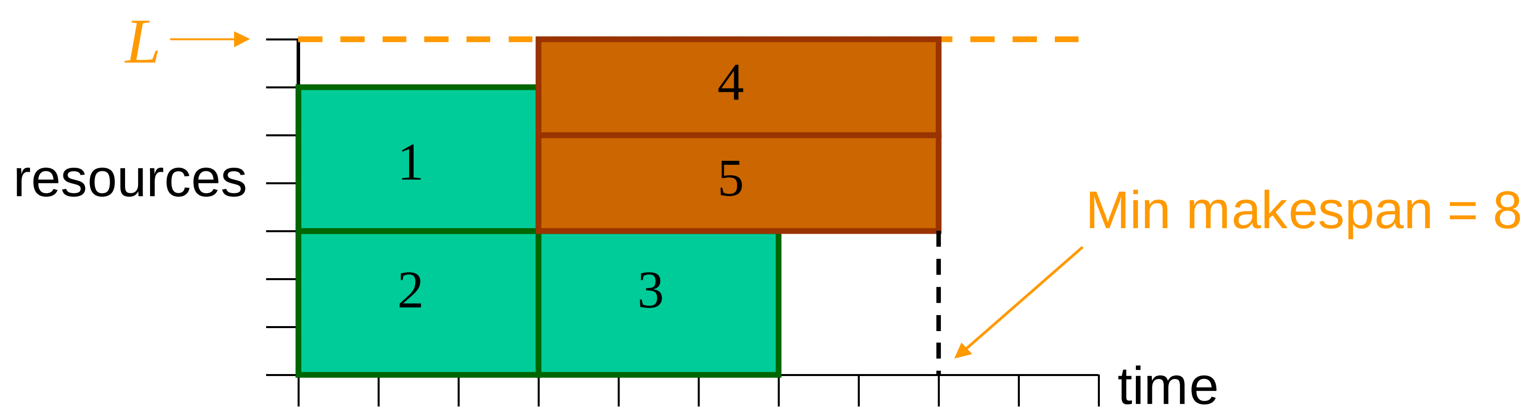
$$t_{j_1} + \cdots + t_{j_k} \geq \sum_{i=1}^{k} \left((k - i + \frac{1}{2}) \frac{r_i}{L} - \frac{1}{2} \right) d_i$$

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:

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

Facet defining

Resulting bound:

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

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:

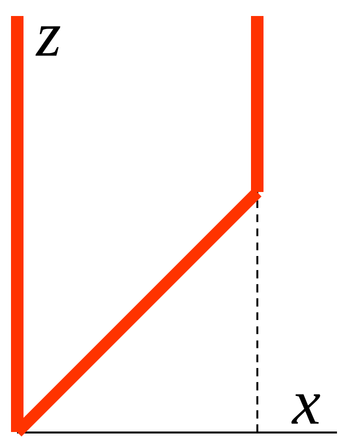
$$\begin{pmatrix} \text{no machine} \\ x = 0 \end{pmatrix} \vee \begin{pmatrix} \text{small machine} \\ z = 50 \\ x \le 5 \end{pmatrix} \vee \begin{pmatrix} \text{large machine} \\ z = 80 \\ x \le 10 \end{pmatrix}$$

Fixed cost of machine

Output of machine

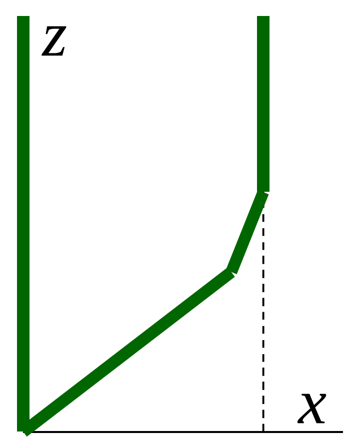
Convex hull relaxation:

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



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.)

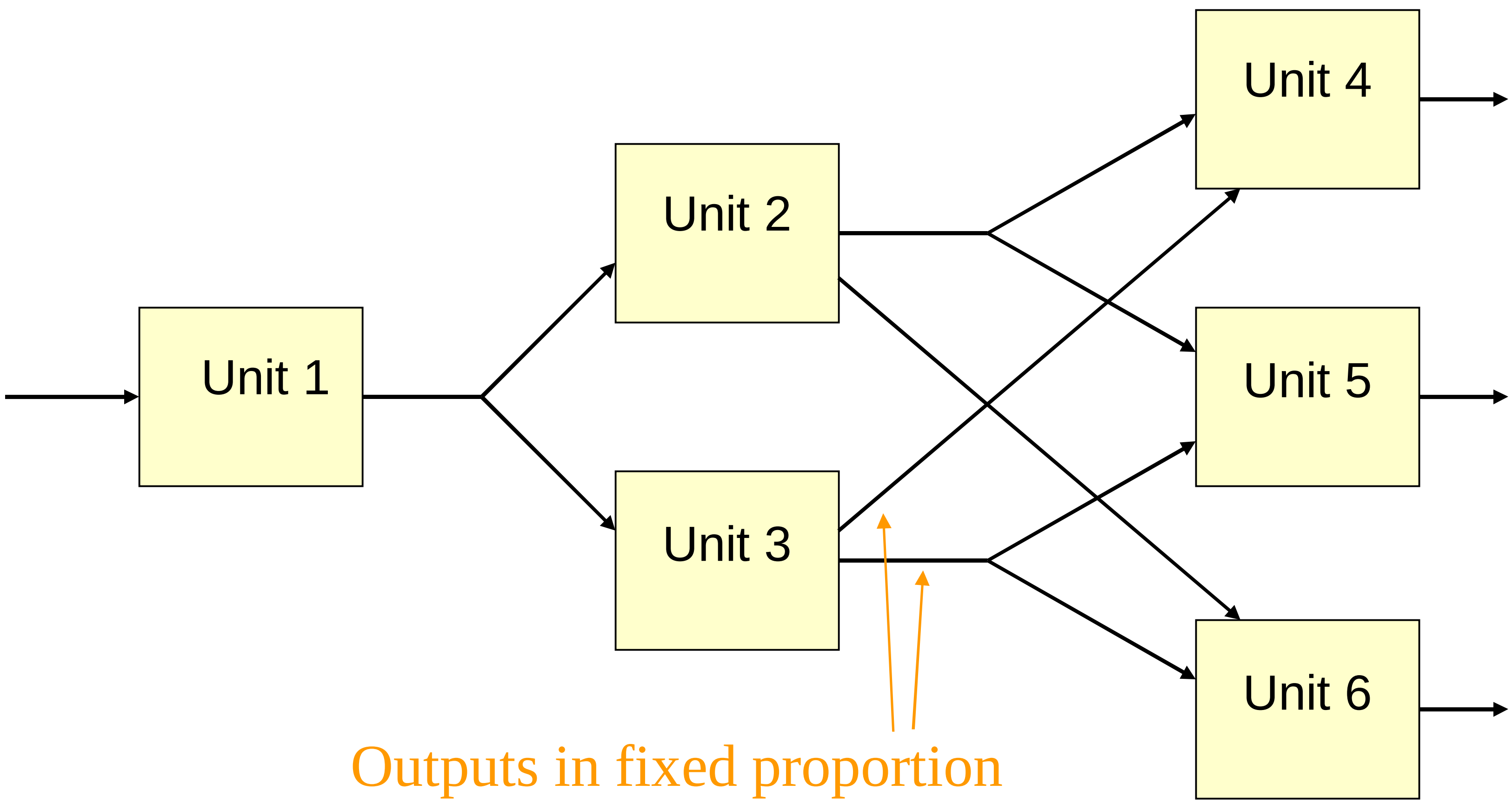
Elements of a General Scheme

- 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

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

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* **14** (2002) 295-321.
- 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