Constraint Programming

Short Course for Air Products & Chemicals

April 2003

John Hooker

Carnegie Mellon University

Carnegie Mellon GSIA

I. Overview and Success Stories (1 hour)

II. Basic Concepts and Problem Formulation (1.5 hours)

III. Algorithmic Ideas (1 hour)

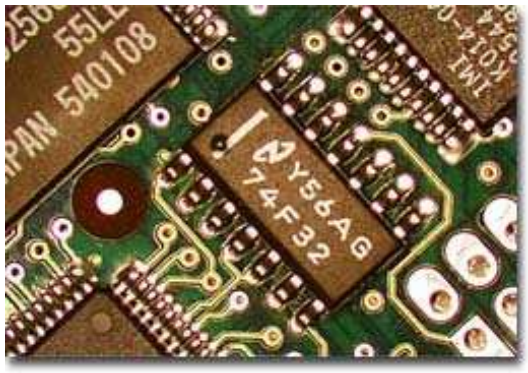
I. Overview and Success Stories

What is Constraint Programming?

- It is a relatively new technology developed in the computer science and artificial intelligence communities.
- It has found an important role in scheduling, logistics and supply chain management.

Early Commercial Successes

- Circuit design (Siemens)



- Container port scheduling (Hong Kong and Singapore)

- Real-time control (Siemens, Xerox)

Applications

- Job shop scheduling
- Assembly line smoothing and balancing
- Cellular frequency assignment
- Nurse scheduling
- Shift planning
- Maintenance planning
- Airline crew rostering and scheduling
- Airport gate allocation and stand planning

Applications

- Production scheduling
  - chemicals
  - aviation
  - oil refining
  - steel
  - lumber
  - photographic plates
  - tires
- Transport scheduling (food, nuclear fuel)
- Warehouse management
- Course timetabling

Constraint Programming vs. Mathematical Programming

• Traditional **mathematical programming** methods are very good at many supply chain problems but often have difficulty with **scheduling and sequencing**.

• **Constraint programming** can excel at **scheduling and sequencing**.

• The two naturally can work together in a supply chain context.

Constraint Programming vs. Mathematical Programming

• **Mathematical programming** methods rely heavily on **numerical calculation**. They include

  • linear programming (LP)

  • mixed integer programming (MIP)

  • nonlinear programming (NLP)

• **Constraint programming** relies heavily on **constraint propagation** (a form of logical inference).

Programming ≠ Programming

- In **constraint programming**:
  - *programming* = a form of computer programming
- In **mathematical programming**:
  - *programming* = planning

Constraint Programming vs. Mathematical Programming

• In **mathematical programming**, equations (constraints) describe the problem but don't tell how to solve it.

• In **computer programming**, a procedure tells how to solve the problem.

• In **constraint programming**, each constraint invokes a procedure that screens out unacceptable solutions.

Major Vendors

COSYTEC

- **CHIP –** General state-of-the-art constraint programming
- **WAREPLAN –** Warehouse management

Major Vendors

- **CPLEX –** Linear programming (LP) and mixed integer/linear programming (MILP)
- **Solver** – General state-of-the-art constraint programming (CP)
- **Scheduler** – CP with special-purpose scheduling constraints
- **OPL Studio** – Modeling framework for both CP and MILP
- **ODF** (Optimization Development Framework) – Tool for developing optimization extensions for SAP's APO.

Major Vendors

dash optimization

- **XPRESS-MP –** Linear, nonlinear and mixed integer programming
- **XPRESS-XL –** Spreadsheet-based optimization
- **Mosel –** Framework for integrating solvers
  - Dash has a cooperative agreement with Cosytec to use CHIP.

Major Vendors

- **ECLiPSe –** High-level modeling language for MILP and CP
  - IC Parc is an extension of Imperial College, London

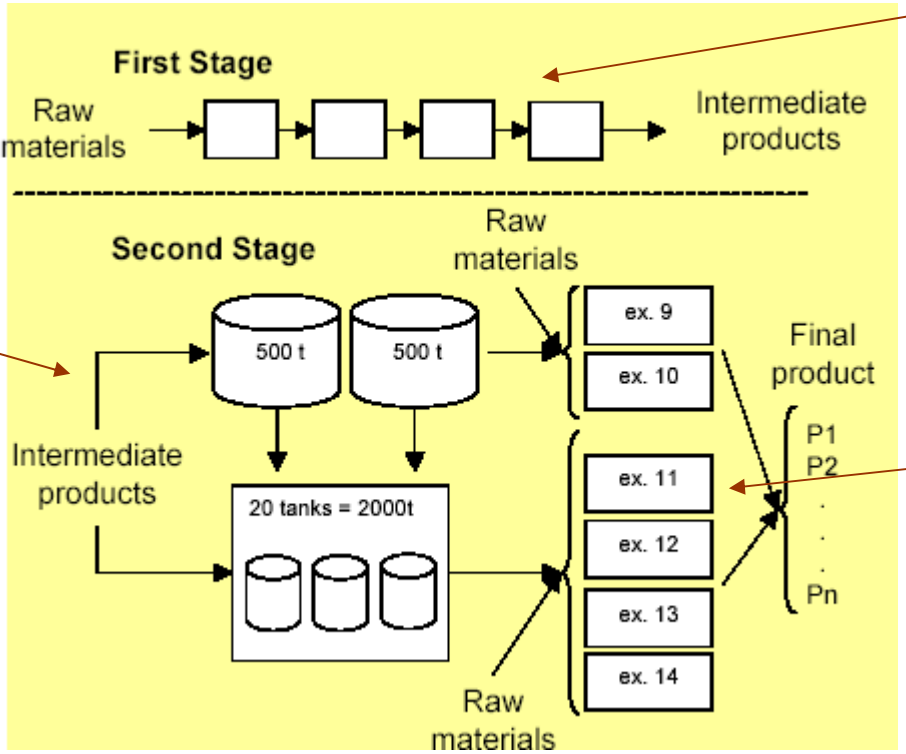
Some Success Stories

- These are chosen because:
  - They are very recent (released 2 weeks ago).
  - They illustrate how scheduling interacts with other aspects of supply chain.
    - And thus how CP can interact with other methods.
  - Since they are part of a government (EU) supported project (LISCOS), a fair amount of detail was released to public.
- All are solved with help of Dash's Mosel system.

Process Scheduling and Lot Sizing at BASF

Manufacture of polypropylenes in 3 stages

polymerization



intermediate storage

extrusion

BASF

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

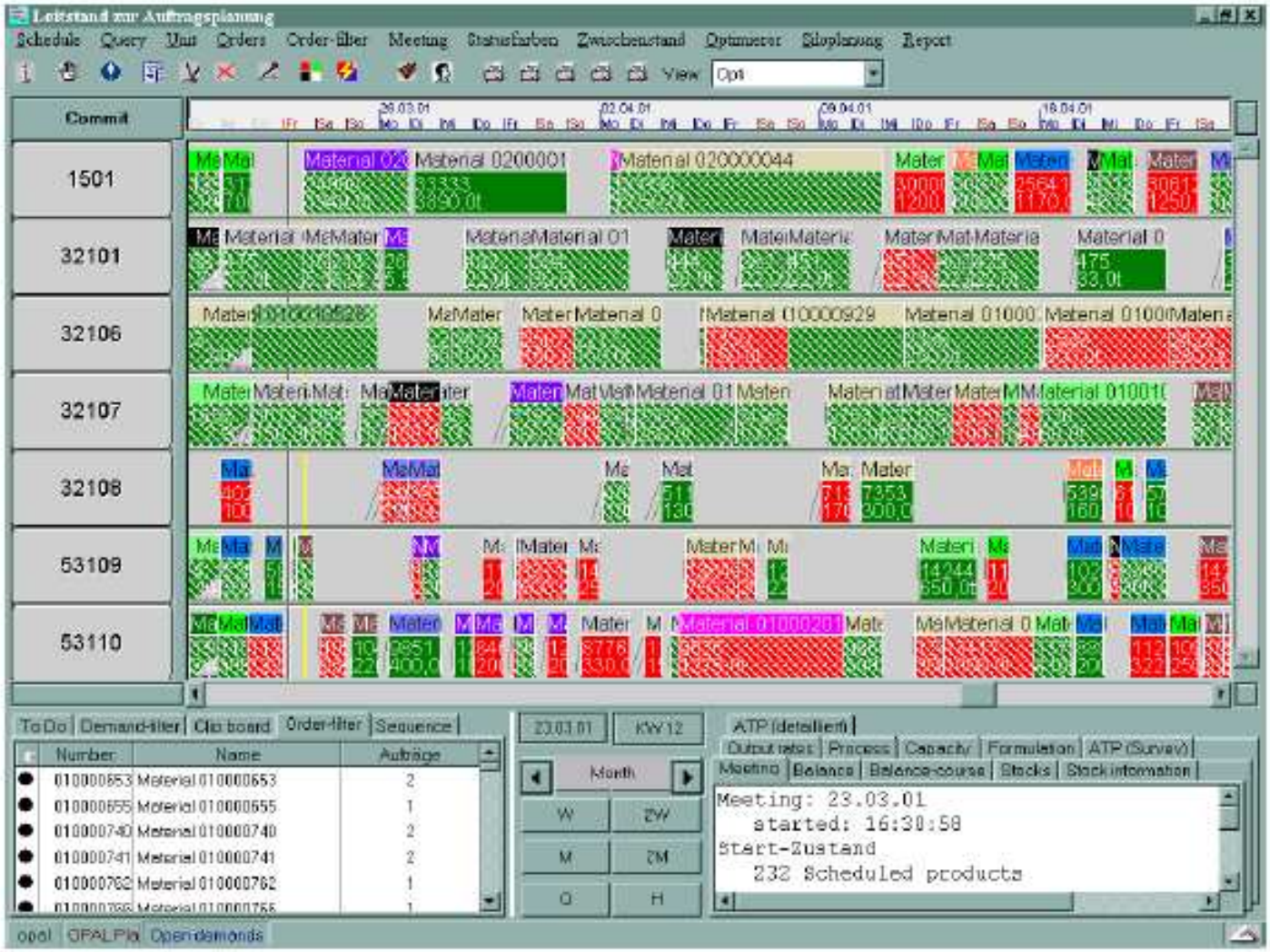
Process Scheduling and Lot Sizing at BASF

- 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)



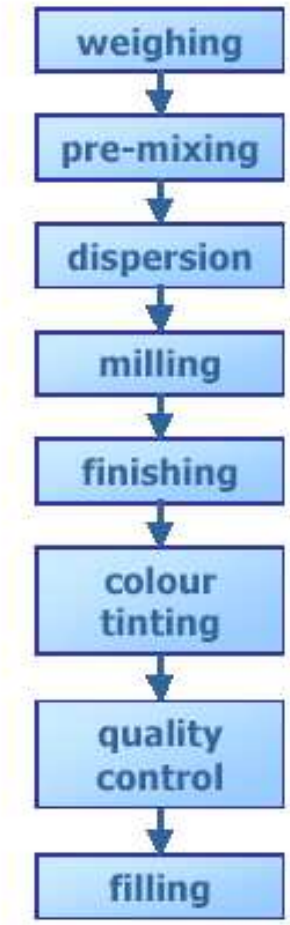
Source: BASF

Carnegie Mellon GSIA

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 Bardot



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

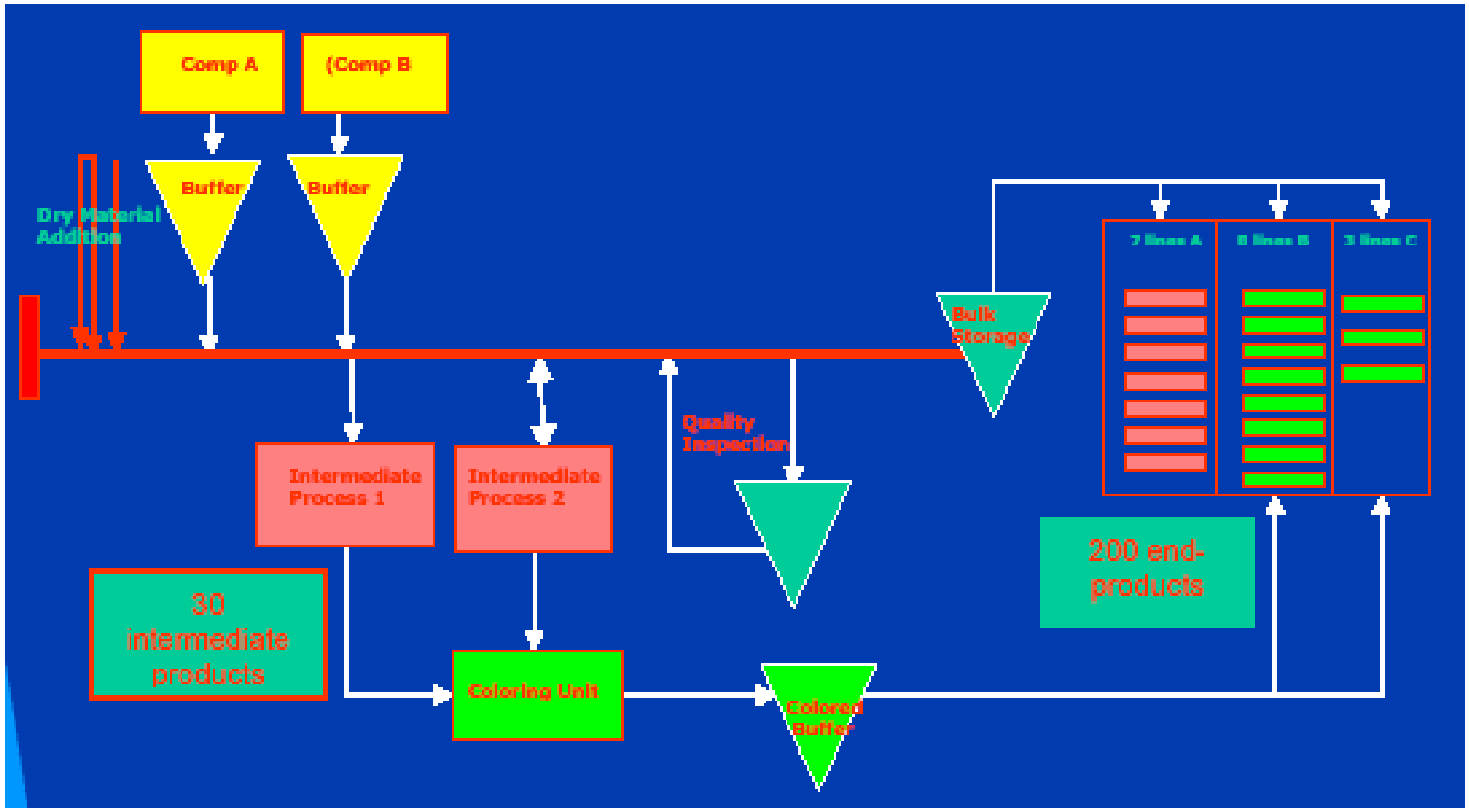
- Solution method similar to BASF case (MIP + CP).
- Benefits
  - Optimal solution obtained in a few minutes for 20 machines and 80 products.
  - Product shortages eliminated.
  - 10% increase in output.
  - Fewer cleanup materials.
  - Customer lead time reduced.

Discrete Lot Sizing at Procter and Gamble

- Continuously running production line.
- Manufactures 50 snack foods in discrete batches.
  - A batch may consume one or more 8-hr periods.
  - At most one batch per period (90-150 periods).
  - Sequencing constraints.

Procter&Gamble

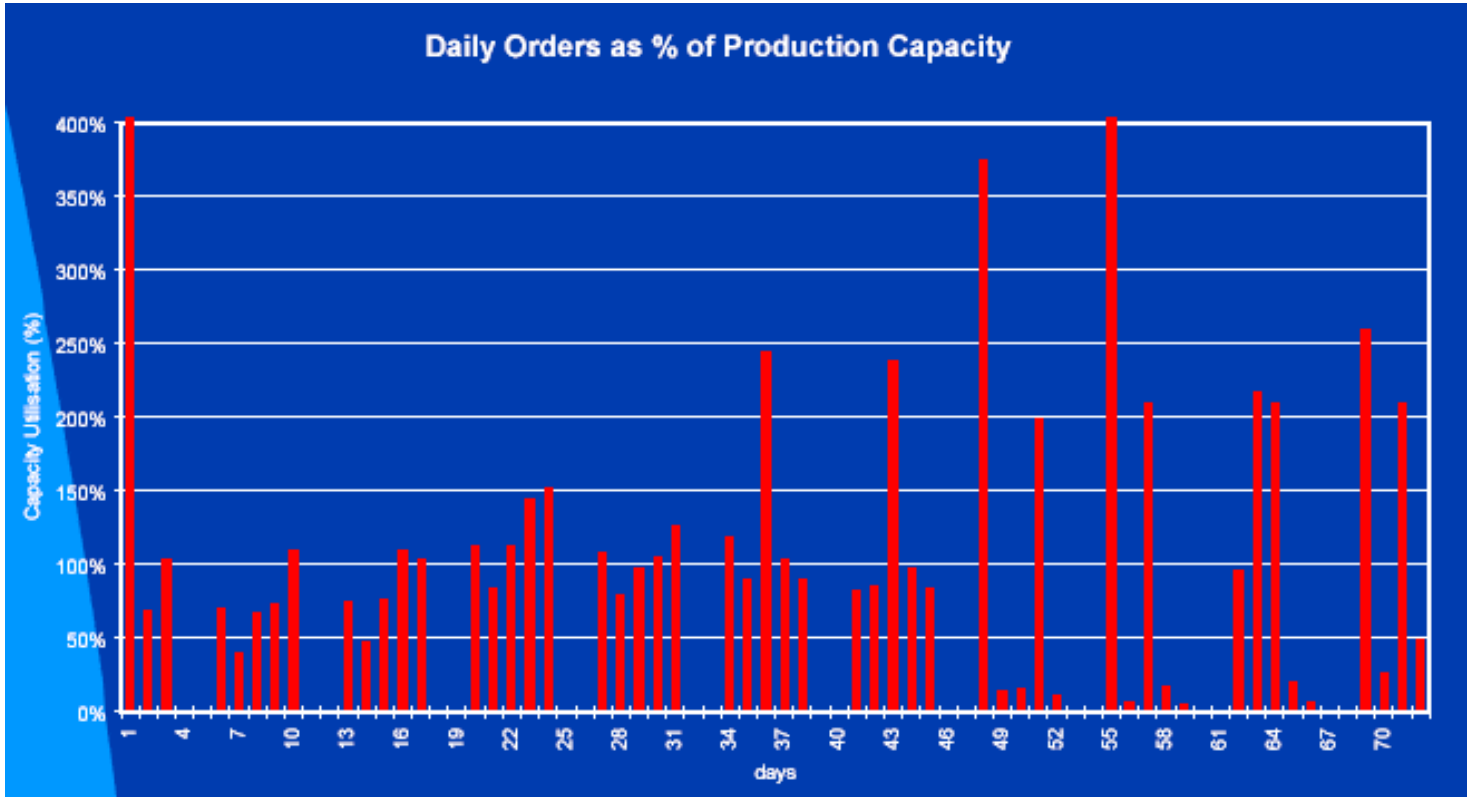
Simplified production process



Source: P&G

Discrete Lot Sizing at P&G

- Highly variable demand



Source: P&G

Discrete Lot Sizing at P&G

- Objective: Choose batch sequencing and sizes so as to:
  - Avoid excessive inventory accumulation.
  - Avoid large batch sizes.
- Solution method did *not* use CP
  - Used a specialized MIP approach.
  - However, this is a natural problem for MIP + CP.

Discrete Lot Sizing at P&G

- Benefits
  - Improved customer service.
  - Complete planning cycle for 150 periods reduced from 4-5 hours to 2 hours.
  - Method is being extended to 13 lines with intermediate storage and packing.

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

Carnegie Mellon GSIA

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.

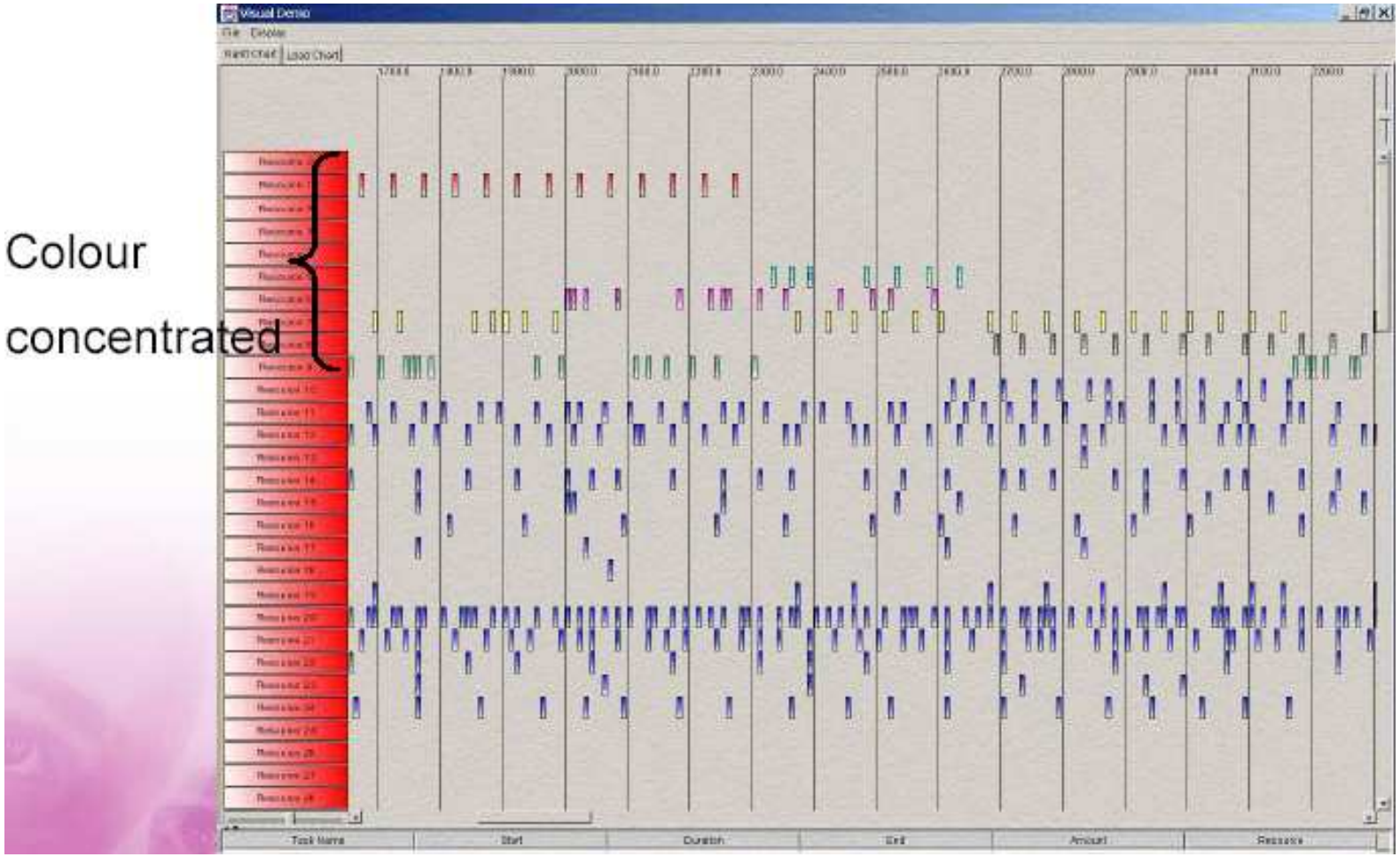
Production Line Sequencing at Peugeot/Citroën

- 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



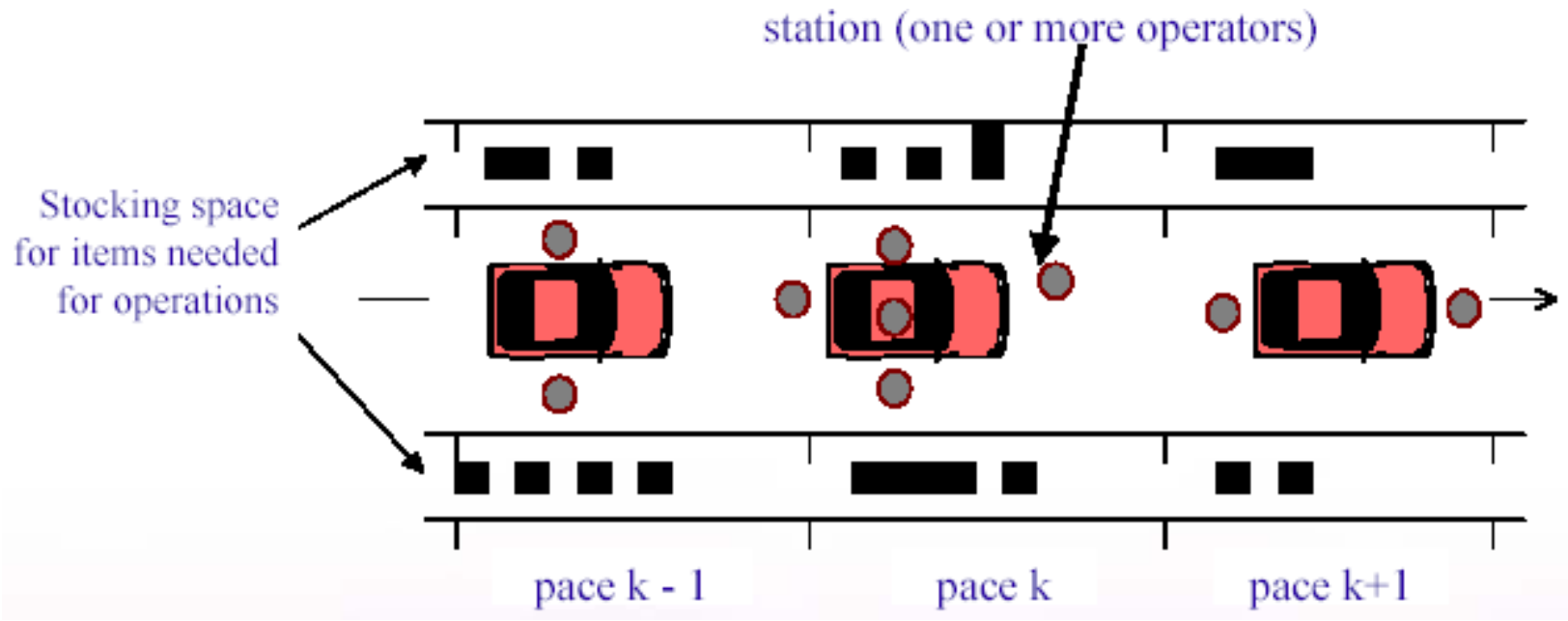
Source: Peugeot/Citroën

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

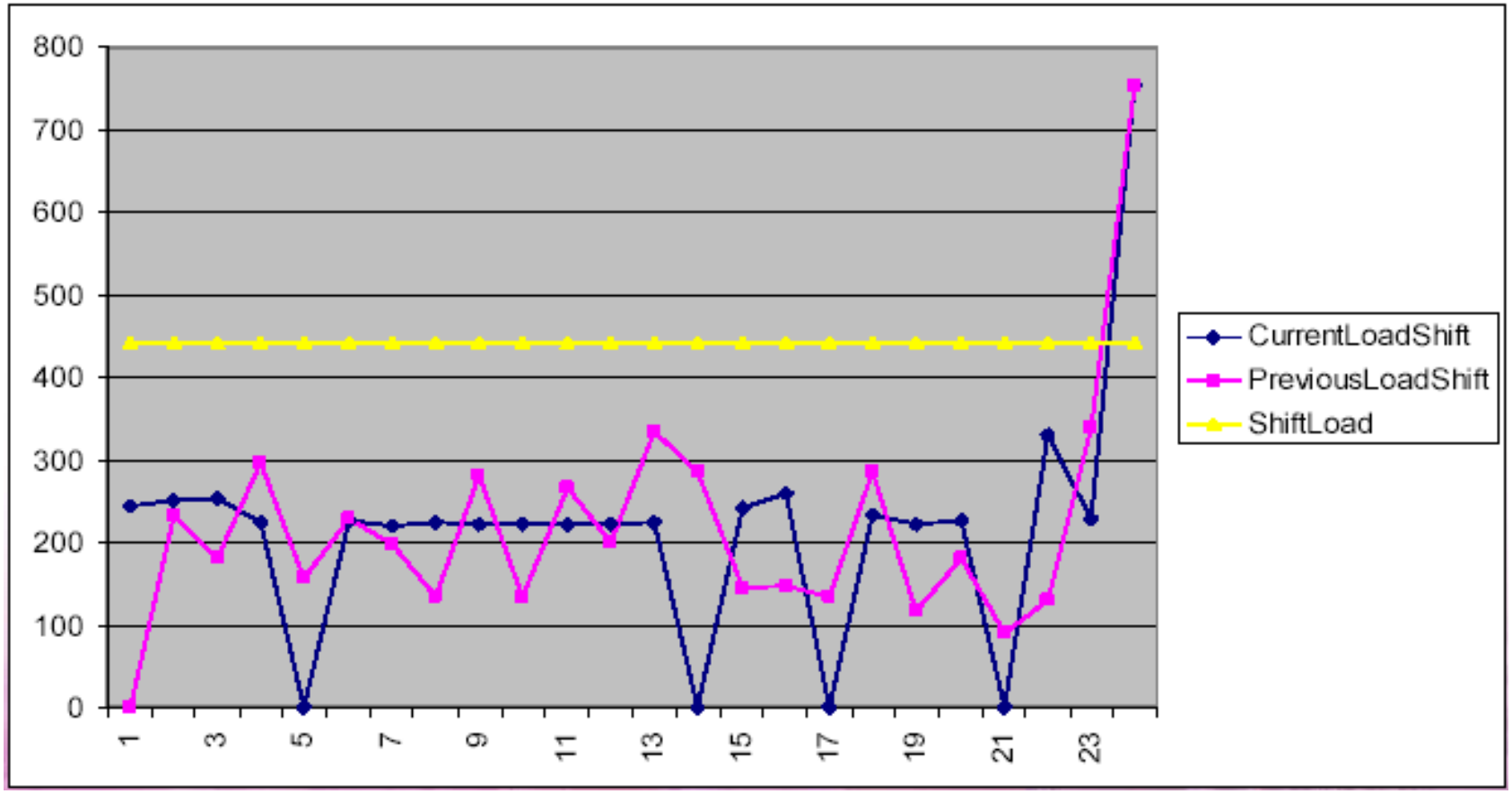
Line Balancing at Peugeot/Citroën

- 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.

Example of load shifting over a typical day



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.

Why Sequencing is Hard

- There are many ways to sequence only a few jobs.
  - 5 jobs can be sequenced 120 ways.
  - 10 jobs can be sequenced 3,628,800 ways.
  - 20 jobs can be sequenced 2,432,902,008,176,640,000 ways.
- Fast computers are useless against this exponential explosion.
- Clever methods can reduce the work, but only up to a point.

Advantages of Constraint Programming

- Usually better at sequencing than other methods.
- Adding messy constraints makes the problem easier.
- More powerful modeling language makes formulation and debugging easier.

Disadvantages

- Cannot deal with cost/profit optimization.
- Not so good at resource allocation, task assignment, inventory control.

Trends

- Constraint programming will become better known to engineering and operations research communities.
- Solvers that fully integrate MIP and CP will be developed (3-5 years).
- Heuristic methods will be integrated with CP.
- MIP/CP/heuristics will be regarded as a single technology.

II. Basic Concepts and Problem Formulation

A Simple Example

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

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

- constraint programming
- integer programming
- a hybrid approach

1. 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.

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

Continuous relaxation

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

Covering inequalities

Relax integrality

Branch and bound (Branch and relax)

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

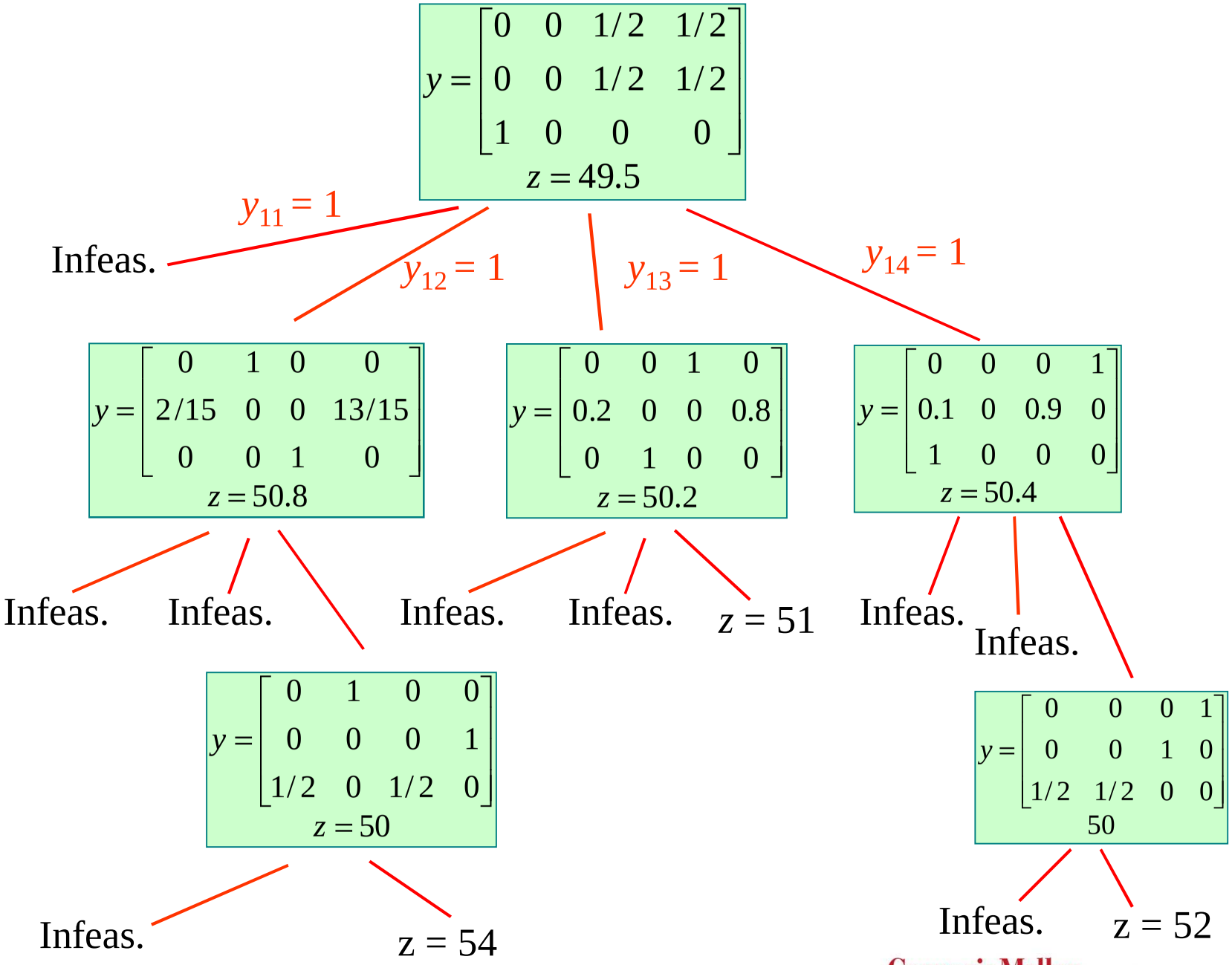
At each node of the branching tree:

- 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.

- Use SOS-1 branching.


$y = \begin{bmatrix} 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 1/2 & 1/2 \\ 1 & 0 & 0 & 0 \end{bmatrix}$
$z = 49.5$
$y_{11} = 1$
Infeas.
$y_{12} = 1$
$y_{13} = 1$
$y_{14} = 1$
$y = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 2/15 & 0 & 0 & 13/15 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
$z = 50.8$
$y = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0.2 & 0 & 0 & 0.8 \\ 0 & 1 & 0 & 0 \end{bmatrix}$
$z = 50.2$
$y = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0.1 & 0 & 0.9 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$
$z = 50.4$
Infeas.
Infeas.
Infeas.
Infeas.
$z = 51$
Infeas.
Infeas.
$y = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1/2 & 0 & 1/2 & 0 \end{bmatrix}$
$z = 50$
$y = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1/2 & 1/2 & 0 & 0 \end{bmatrix}$
50
Infeas.
z = 54
Infeas.
z = 52

2. Solve as a constraint programming problem

Search: Domain splitting
Inference: Domain reduction
Relaxation: Constraint store (set of current variable domains)

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

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

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 $5x_1 + 8x_2 + 4x_3 \le z$
  $3x_1 + 5x_2 + 2x_3 \ge 30$

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

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

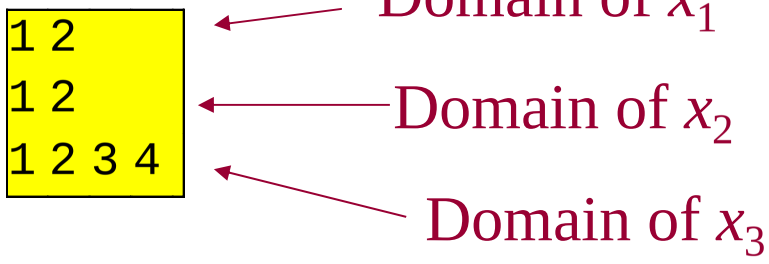
So the domain of x_2 is reduced to $\{2,3,4\}$.

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

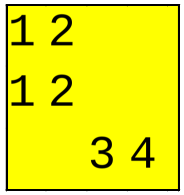
- Maintain hyperarc consistency on

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

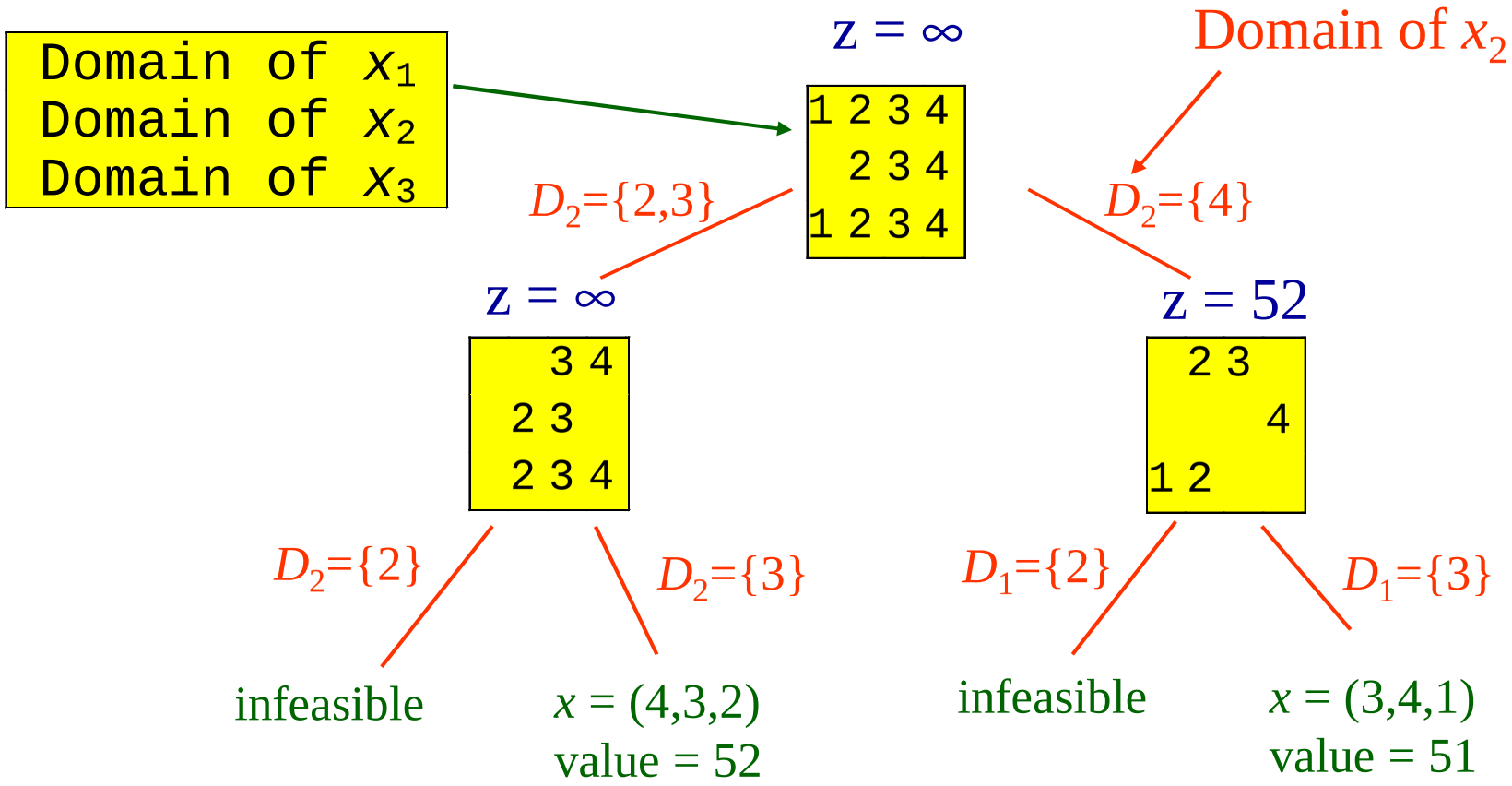
Suppose for example:



Then one can reduce the domains:



- 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

3. 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.

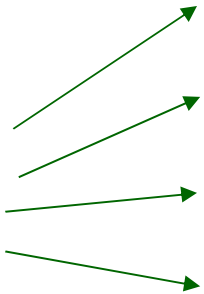
Relaxation:

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

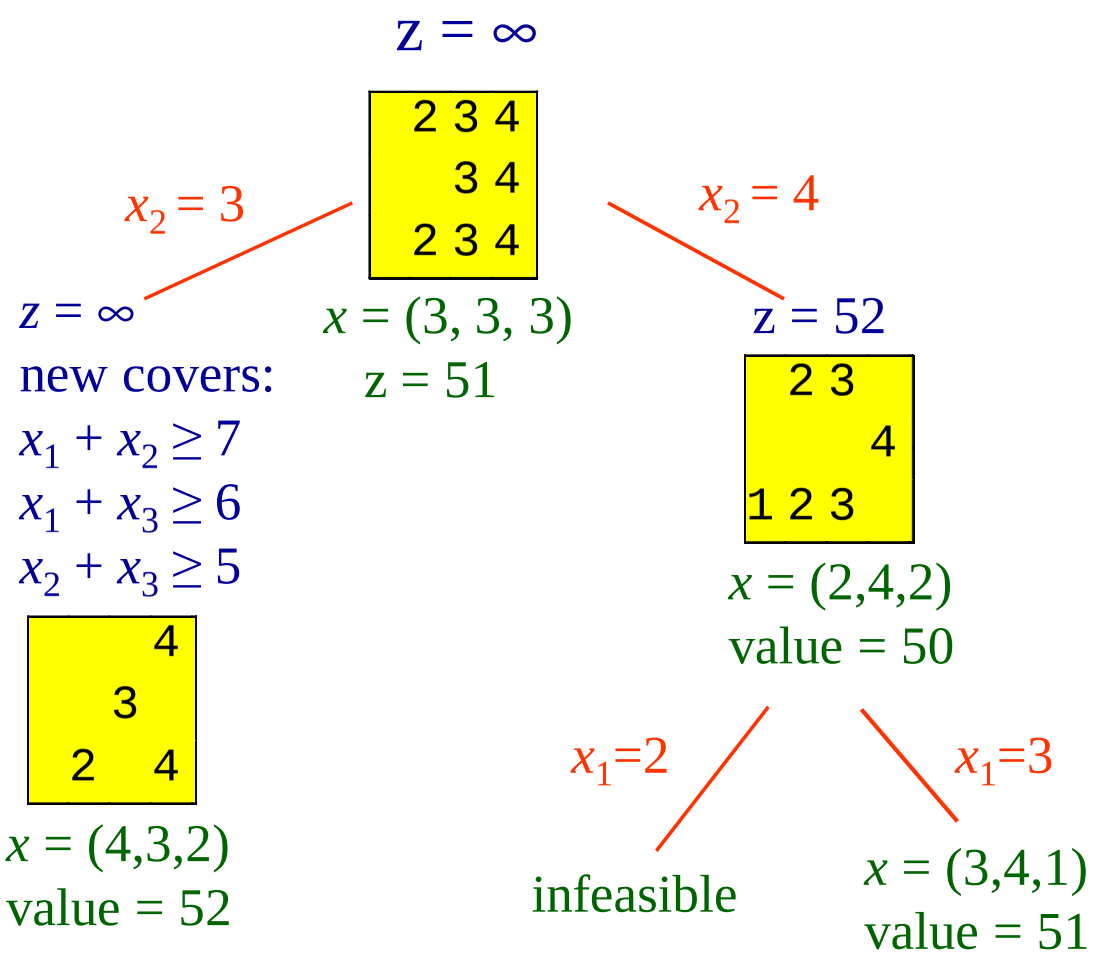
Model for hybrid approach

$$\begin{aligned}
\min \quad & 5x_1 + 8x_2 + 4x_3 \le z \\
\text{s.t.} \quad & 3x_1 + 5x_2 + 2x_3 \ge 30 \\
& \text{all-different}\{x_1, x_2, x_3\} \\
& x_1 + x_2 \ge 6 \\
& x_1 + x_3 \ge 4 \\
& x_2 + x_3 \ge 5 \\
& x_1 + x_2 + x_3 \ge 8 \\
& x_j \in \{1,\ldots,4\}
\end{aligned}$$

Covering inequalities generated with all-diff



Generate and propagate covering inequalities at each node of search tree


z = ∞
2 3 4
3 4
2 3 4
$x_2 = 3$
$x_2 = 4$
z = ∞
new covers:
$x_1 + x_2 \geq 7$
$x_1 + x_3 \geq 6$
$x_2 + x_3 \geq 5$
x = (3, 3, 3)
z = 51
z = 52
2 3
4
1 2 3
x = (2,4,2)
value = 50
4
3
2 4
x = (4,3,2)
value = 52
x_1=2
x_1=3
infeasible
x = (3,4,1)
value = 51

Optimization and Constraint Programming Compared

- Constraint types.
  - Optimization may be superior when constraints contain many variables (and have good relaxations).
  - Constraint programming may be superior when constraints contain few variables (and propagate well).

Optimization and CP Compared

- Exploiting structure
  - Optimization relies on deep analysis of the mathematical structure of specific classes of problems, particularly polyhedral analysis, which yields strong cutting planes.
  - Constraint programming identifies subsets of problem constraints that have special structure (e.g., `all-different, cumulative`) and apply tailor-made domain-reduction algorithms.

Optimization and CP Compared

- Relaxation and inference.
  - Optimization creates strong relaxations with cutting planes, Lagrangean relaxation, etc. These provide bounds on the optimal value.
  - Constraint programming exploits the power of inference, especially in domain reduction algorithms. This reduces the search space.

Optimization and CP Compared

- Modeling style
  - Optimization uses declarative models that can be solved with a variety of algorithms. But the language is highly restricted (e.g, inequality constraints).
  - Constraint programming models are formulated in a quasi-procedural manner that gives the user more opportunity to direct the solution algorithm. But the model is more closely tied to the solution method.

Consistency

• A constraint set is **consistent** if every partial assignment that violates no constraint is feasible (i.e., can be extended to a feasible solution).

- Consistency is not the same as feasibility.
- Consistency means that all infeasible partial assignments are explicitly ruled out by a constraint.
- Fully consistent constraint sets can be solved without backtracking.

General Consistency

Consider the constraint set

$$x_1 + x_{100} \geq 1$$
$$x_1 - x_{100} \geq 0$$
$$x_j \in \{0,1\}$$

It is not consistent, because $x_1 = 0$ violates no constraint and yet is infeasible (no solution has $x_1 = 0$).

Adding the constraint $x_1 = 0$ makes the set consistent.

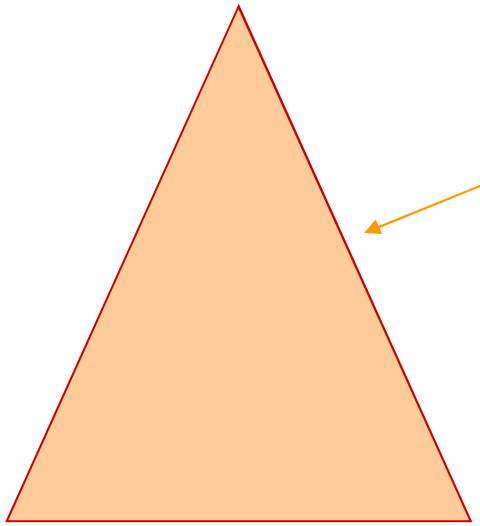
$x_1 + x_{100} \geq 1$

$x_1 - x_{100} \geq 0$

other constraints

$x_j \in \{0,1\}$

$x_1 = 0$ $\qquad$ $x_1 = 1$



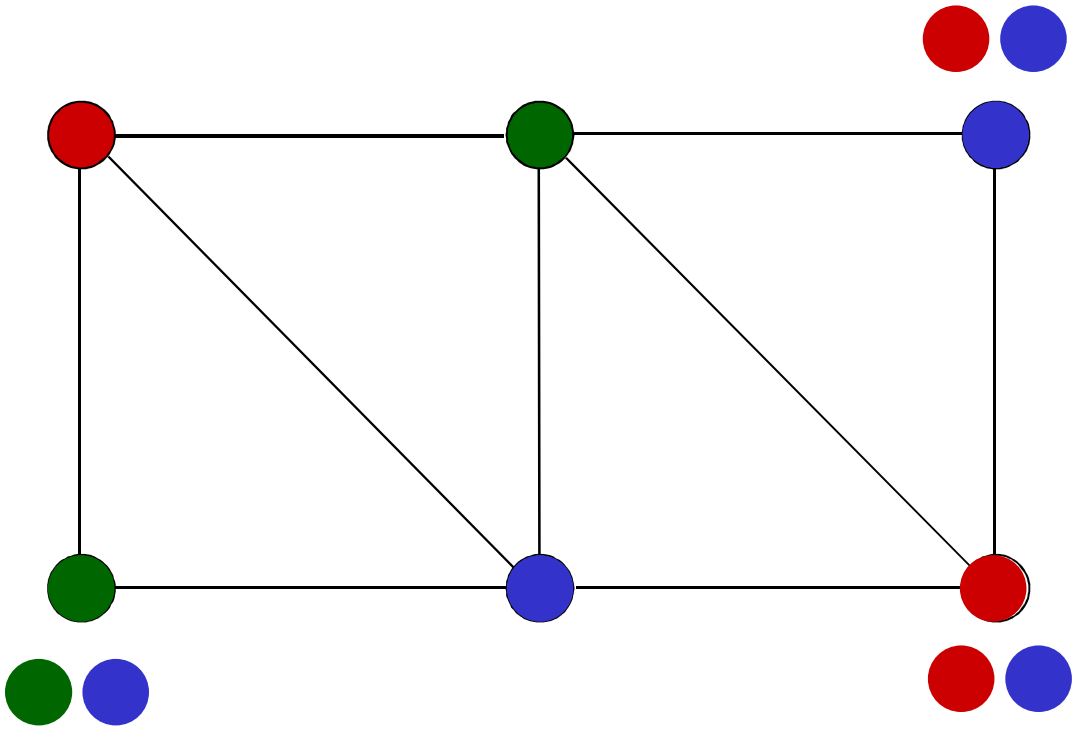
subtree with 2^{99} nodes but no feasible solution

By adding the constraint $x_1 = 0$, the left subtree is eliminated

Hyperarc Consistency

- A constraint set is **hyperarc consistent** if every value in every variable domain is part of some feasible solution.
  - That is, the domains are reduced as much as possible.
  - If all constraints are “binary” (contain 2 variables), hyperarc consistent = arc consistent.
  - Domain reduction is CP’s biggest engine.

Graph coloring problems that can be solved by arc consistency maintenance alone.



A Modeling Example

Traveling salesman problem:

Let c_{ij} = distance from city i to city j.

Find the shortest route that visits each of n cities exactly once.

Integer Programming Model

Let x_{ij} = 1 if city i immediately precedes city j, 0 otherwise

$$\text{minimize} \quad \sum_{ij} c_{ij} x_{ij}$$

$$\text{subject to} \quad \sum_{i} x_{ij} = 1, \quad \text{all } j$$

$$\sum_{j} x_{ij} = 1, \quad \text{all } i$$

$$\sum_{i \in V} \sum_{j \in W} x_{ij} \geq 1, \quad \text{all disjoint } V, W \subset \{1, \ldots, n\}$$

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

Subtour elimination constraints

Constraint Programming Model

Let y_k = the kth city visited.

The model would be written in a specific constraint programming language but would essentially say:

Variable indices

$$\text{minimize} \quad \sum_k c_{y_k y_{k+1}}$$

$$\text{subject to} \quad \text{all-different}(y_1, \ldots, y_n)$$

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

"Global" constraint

Multiple Formulations

Assignment problem with two linked formulations

min	some objective
s.t.	constraints on x_i's
	constraints on y_j's
	$x_{y_j} = j$, all j

x_i = employee assigned time slot i
y_j = time slot assigned employee j

Multiple Formulations

- The linkage of two models improves constraint propagation.
- Here a variable (rather than a constant) has a variable index.

Element constraint

The constraint $c_y \leq 5$ can be implemented:

$z \leq 5$
$\text{element}(y, (c_1, \ldots, c_n), z)$

Assign z the yth value in the list

The constraint $x_y \leq 5$ can be implemented:

$z \leq 5$
$\text{element}(y, (x_1, \ldots, x_n), z)$

Add the constraint $z = x_y$

(this is a slightly different constraint)

Cumulative constraint

- Used for resource-constrained scheduling.
- Total resources consumed by jobs at any one time must not exceed L.

$$\text{cumulative}\big((t_1,\ldots,t_n),(d_1,\ldots,d_n),(r_1,\ldots,r_n),L\big)$$

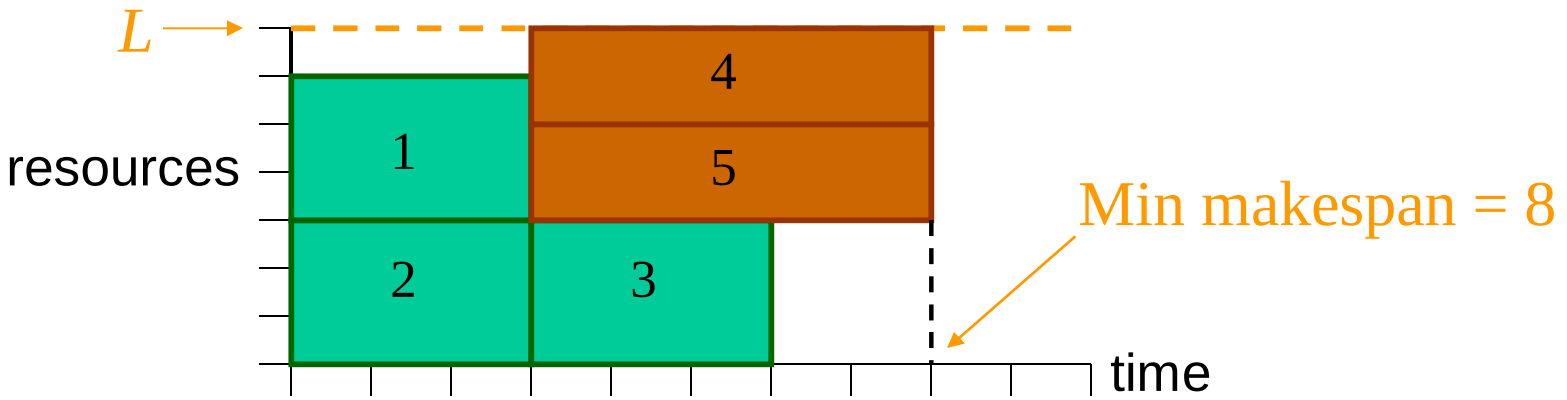
Job start times

Job durations

Job resource requirements

Cumulative Constraint

Minimize makespan (no deadlines, all release times = 0):



$$\begin{aligned} \min \quad & z \\ \text{s.t.} \quad & \text{cumulative}\big((t_1,\ldots,t_5),(3,3,3,5,5),(3,3,3,2,2),7\big) \\ & z \geq t_1 + 3 \\ & \vdots \\ & z \geq t_5 + 2 \end{aligned}$$

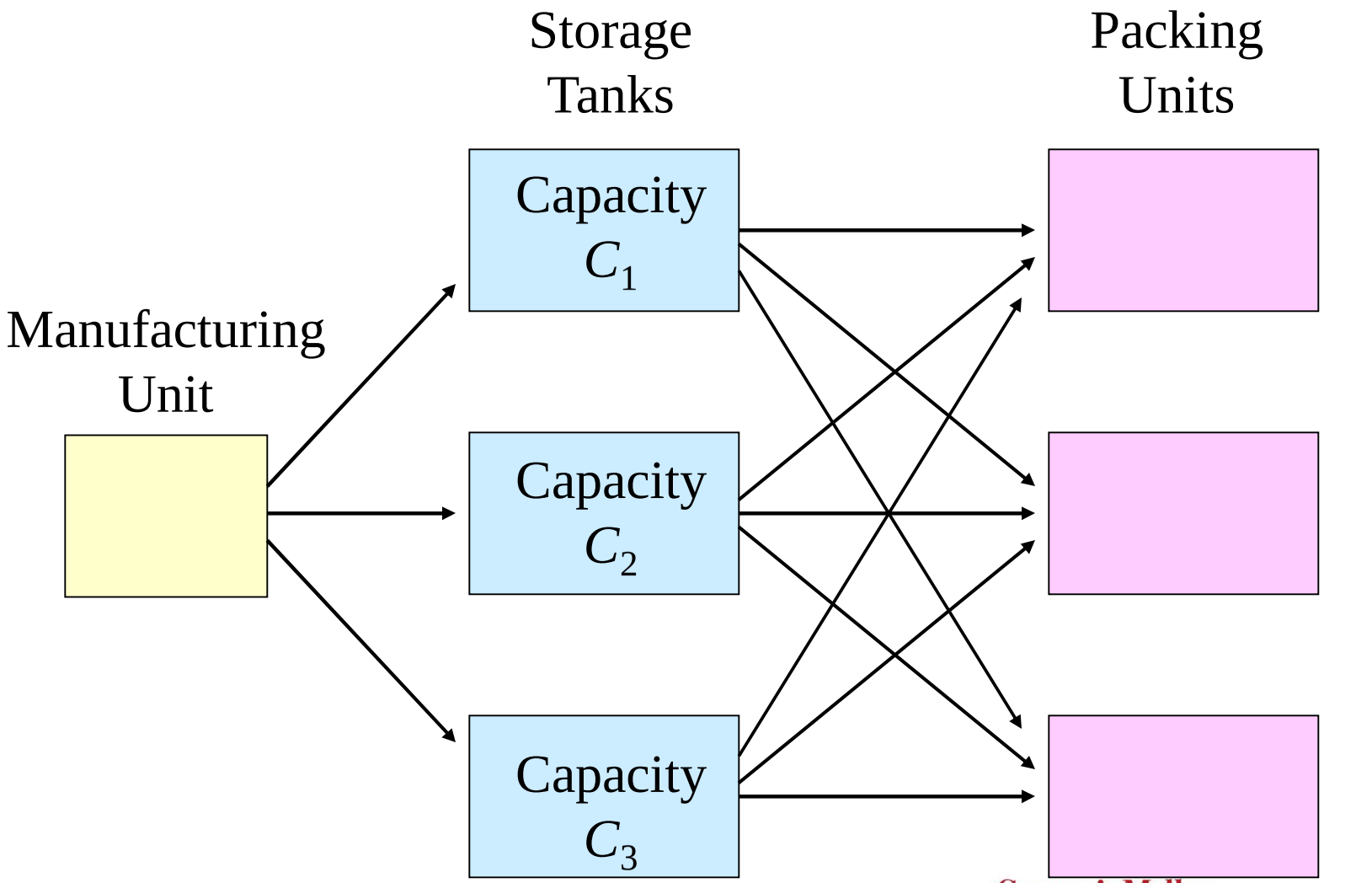
L

Resources used

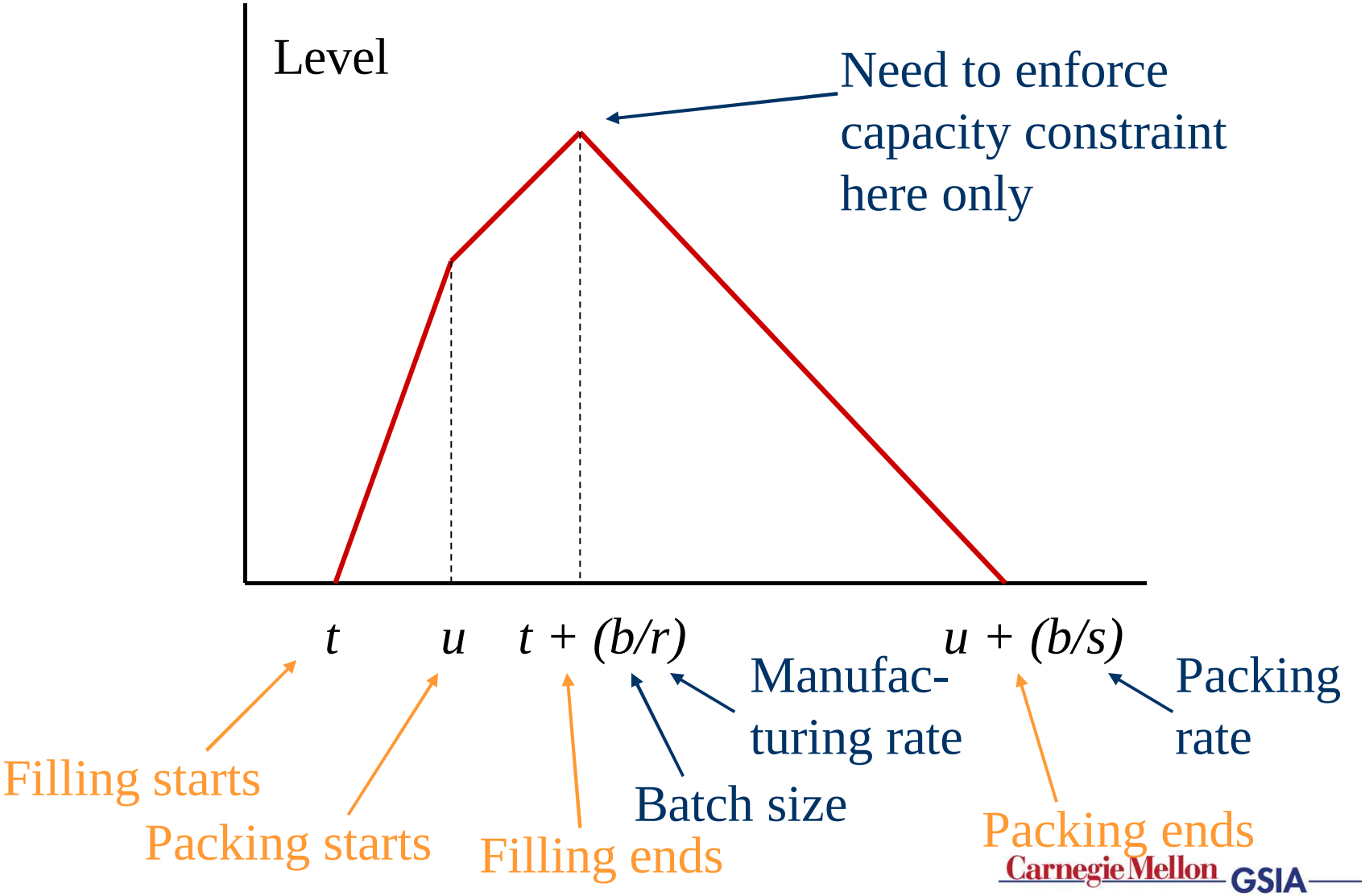
Durations

Job start times

Production Scheduling



Filling of StorageTank



$$\min \; T$$ ← Makespan

$$\text{s.t.} \quad T \geq u_j + \frac{b_j}{s_j}, \quad \text{all } j$$

$$t_j \geq R_j, \quad \text{all } j$$ ← Job release time

$$\text{cumulative}(t, v, (1, \ldots 1), m)$$ ← m storage tanks

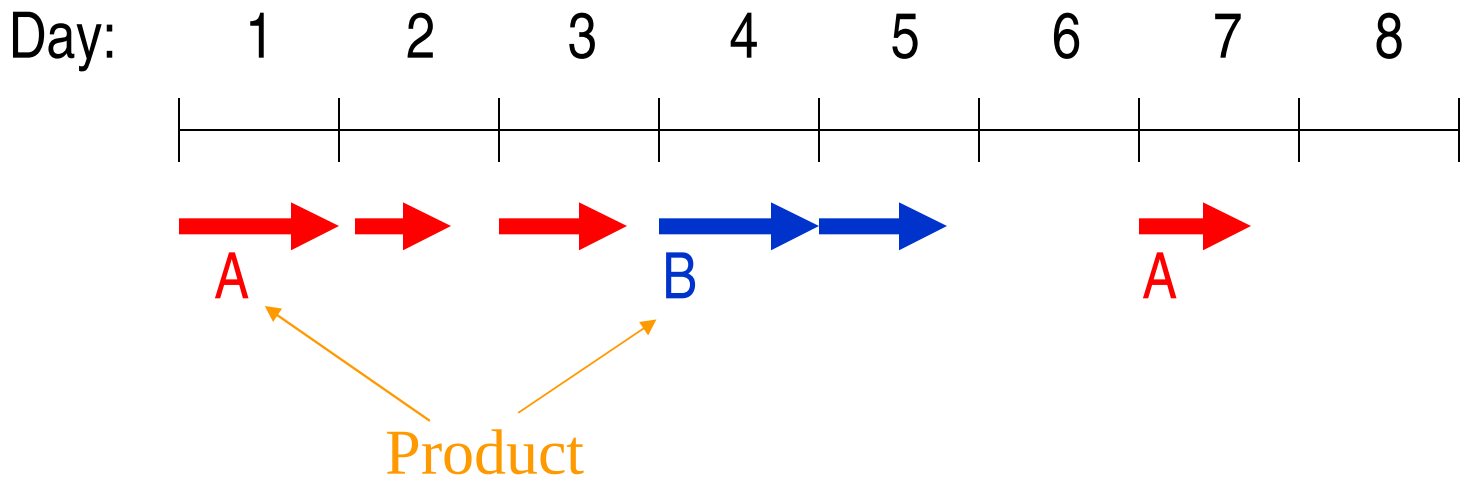
$$v_i = u_i + \frac{b_i}{s_i} - t_i, \quad \text{all } i$$ ← Job duration

$$b_i\left(1 - \frac{s_i}{r_i}\right) + s_i u_i \leq C_i, \quad \text{all } i$$ ← Tank capacity

$$\text{cumulative}\left(u, \left(\frac{b_1}{s_1}, \ldots, \frac{b_n}{s_n}\right), e, p\right)$$ ← p packing units

$$u_j \geq t_j \geq 0$$

Example of Model Simplification: Lot sizing & scheduling



- At most one product manufactured on each day.
- Demands for each product on each day.
- Minimize setup + holding cost.

Integer programming model

(Wolsey)

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

Many variables

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

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

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

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

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

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

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

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

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

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

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

Hybrid model

Minimize holding and setup costs

$$\min \quad \sum_t \left(q_{y_{t-1} y_t} + \sum_i h_i s_{it} \right)$$

Inventory balance

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

Production capacity

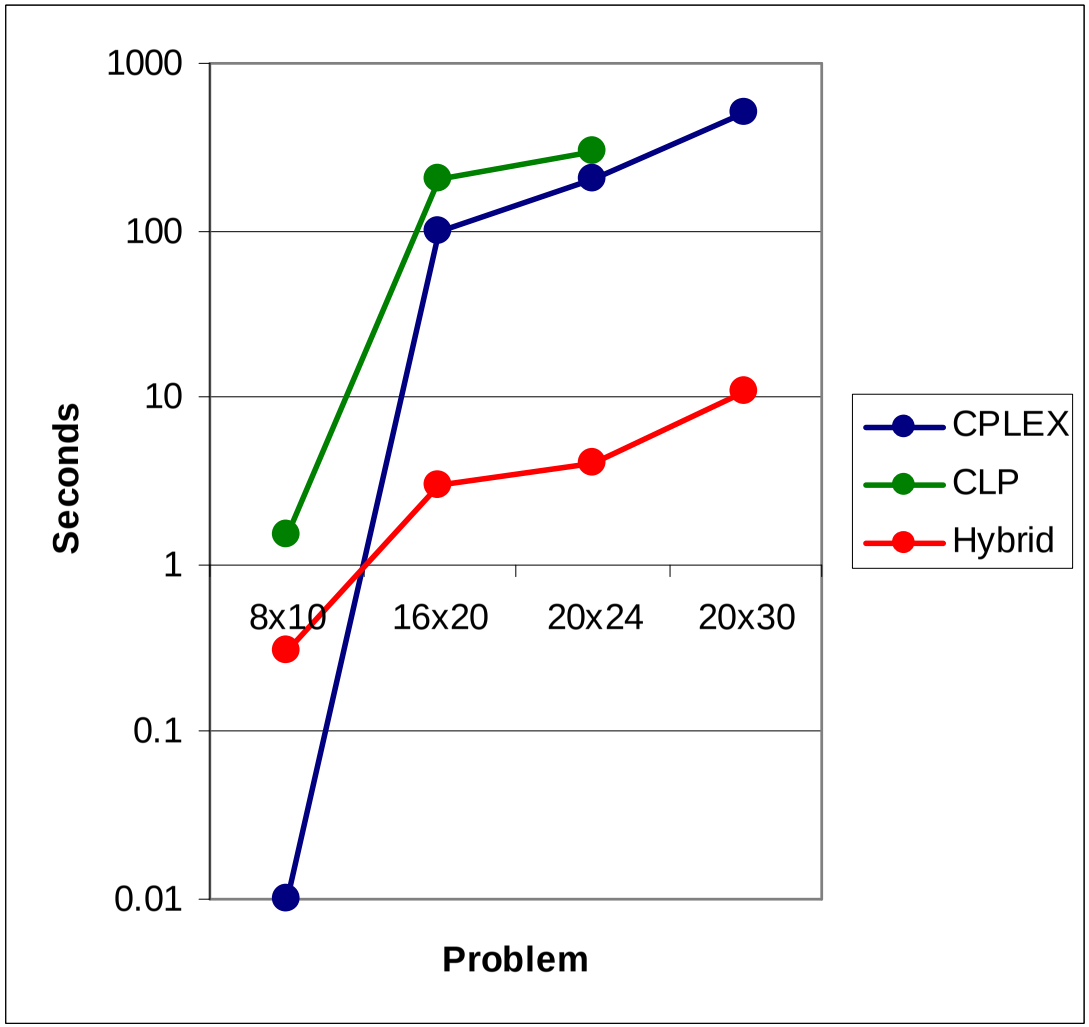
$$0 \le x_{it} \le C, \ s_{it} \ge 0, \ \text{all } i,t$$

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

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.

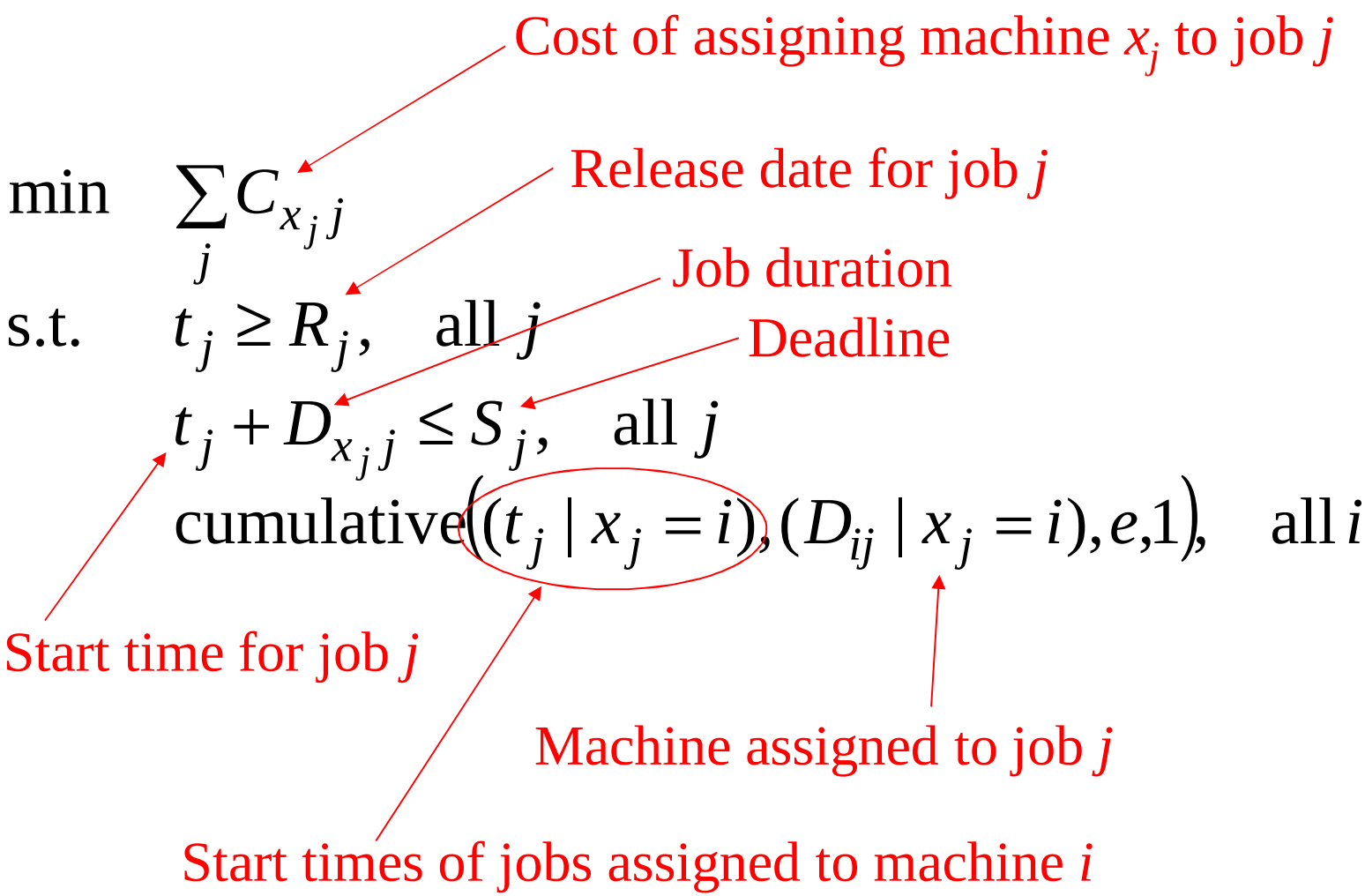
Computational Results *(Ottosson & Thorsteinsson)*



Example of Faster Solution: Machine scheduling

- Schedule jobs on machines that run at different speeds and incur different costs.
- Each job has a release time and deadline.
- Minimize cost using logic-based Benders decomposition.

A model for the problem:



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

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

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 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 →

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

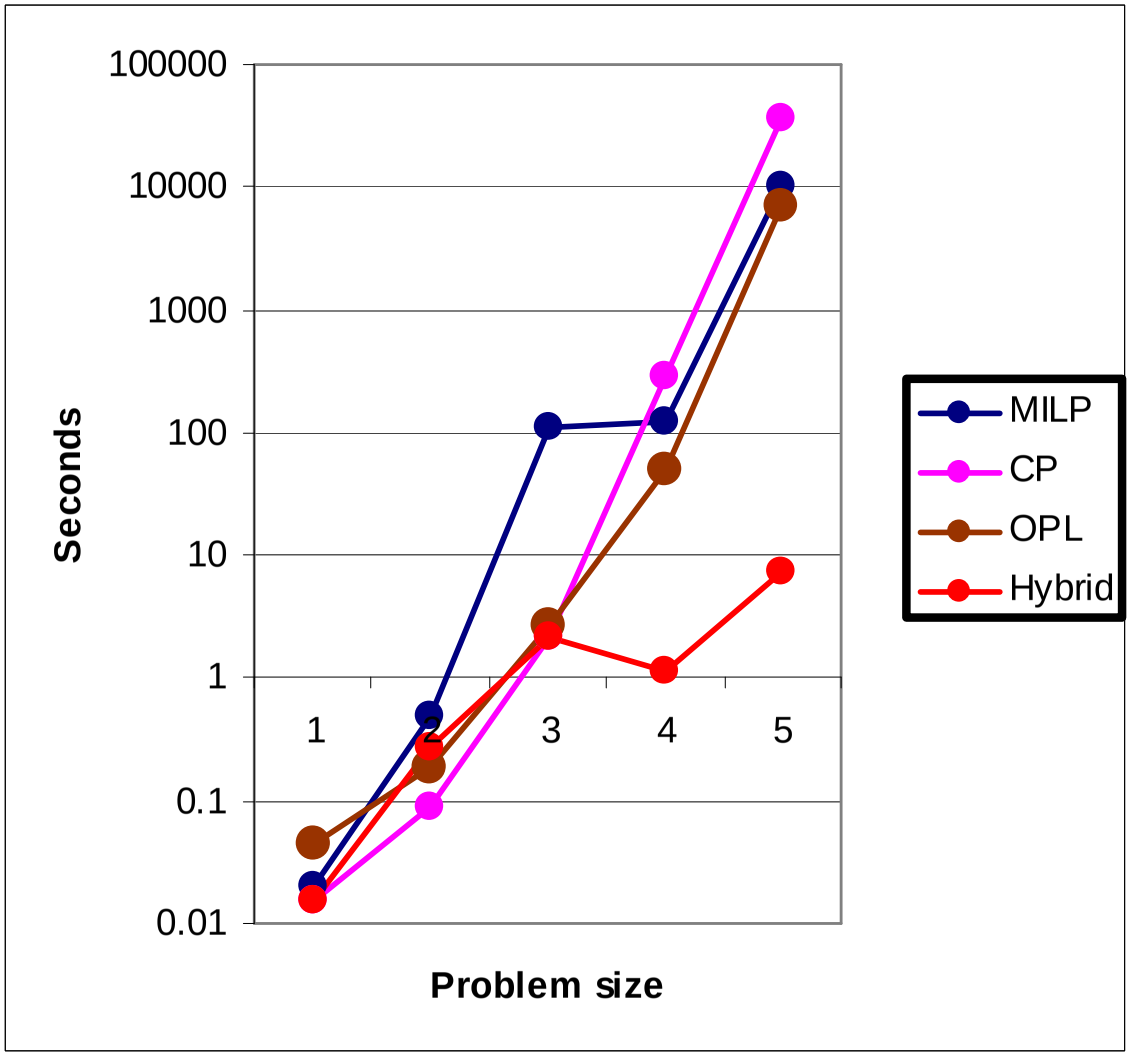
$$y_{ij} \in \{0,1\}$$

Computational Results *(Jain & Grossmann)*

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

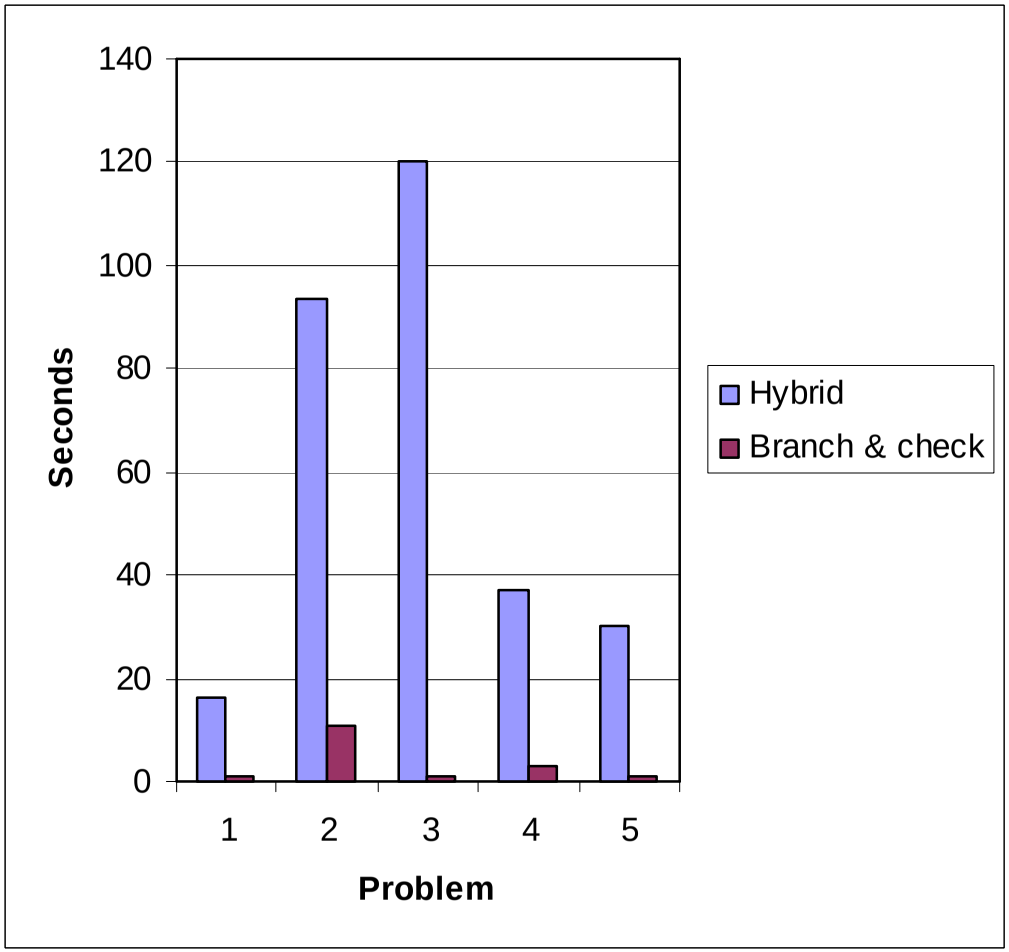
Each data point represents an average of 2 instances

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



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

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



CP Modeling Example

- Will use ILOG's OPL Studio modeling language.
- Ship loading problem
  - Load 34 items on the ship in minimum time (min makespan)
  - Each item requires a certain time and certain number of workers.
  - Total of 8 workers available.
  - Example is from OPL Studio 3.5 Language Manual, pp 308-310.

Item	Dura-tion	Labor
1	3	4
2	4	4
3	4	3
4	6	4
5	5	5
6	2	5
7	3	4
8	4	3
9	3	4
10	2	8
11	3	4
12	2	5
13	1	4
14	5	3
15	2	3
16	3	3
17	2	6

Item	Dura-tion	Labor
18	2	7
19	1	4
20	1	4
21	1	4
22	2	4
23	4	7
24	5	8
25	2	8
26	1	3
27	1	3
28	2	6
29	1	8
30	3	3
31	2	3
32	1	3
33	2	3
34	2	3

Problem data

Precedence constraints

1 → 2,4
2 →3
3 →5,7
4 →5
5 →6
6 →8
7 →8
8 →9
9 →10
9 →14
10 →11
10 →12
11 →13
12 →13
13 →15,16
14 →15
15 →18
16 →17
17 →18
18 →19
18 →20,21
19 →23
20 → 23
21 → 22
22 →23
23 →24
24 →25
25 →26,30,31,32
26 → 27
27 → 28
28 → 29
30 → 28
31 → 28
32 → 33
33 → 34

This is actually a problem that uses the cumulative constraint.

$$\begin{aligned} \min \quad & z \\ \text{subject to} \quad & z \geq t_1 + 3, \;\; z \geq t_2 + 4, \text{ etc.} \\ & \text{cumulative}\big((t_1, \ldots t_{34}), (3,4,\ldots,2), (4,4,\ldots,3), 8\big) \\ & t_2 \geq t_1 + 3, \quad t_4 \geq t_1 + 3, \text{ etc.} \end{aligned}$$

```
int capacity = 8;
int nbTasks = 34;
range Tasks 1..nbTasks;
int duration[Tasks] = [3,4,4,6,…,2];
int totalDuration =
     sum(t in Tasks) duration[t];
int demand[Tasks] = [4,4,3,4,…,3];
struct Precedences {
   int before;
   int after;

}
{Precedences} setOfPrecedences = {
     <1,2>, <1,4>, …, <33,34> };
```

```
scheduleHorizon = totalDuration;
Activity a[t in Tasks](duration[t]);
DiscreteResource res(8);
Activity makespan(0);
minimize
   makespan.end
subject to
   forall(t in Tasks)
      a[t] precedes makespan;
   forall(p in setOfPrecedences)
      a[p.before] precedes a[p.after];
   forall(t in Tasks)
      a[t] requires(demand[t]) res;
};
```

III. Algorithmic Ideas

Domain Reduction for Element

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

can be processed with a discrete domain reduction algorithm that maintains hvperarc consistency.

Domain of z

$$D_z \leftarrow D_z \cap \bigcup_{j \in D_y} D_{x_j}$$

$$D_y \leftarrow D_y \cap \{ j \mid D_x \cap D_{x_j} \neq \emptyset \}$$

$$D_{x_j} \leftarrow \begin{cases} D_z & \text{if } D_y = \{j\} \\ D_{x_j} & \text{otherwise} \end{cases}$$

Example... element$(y,(x_1,x_2,x_3,x_4),z)$

The initial domains are:

$D_z = \{20,30,60,80,90\}$

$D_y = \{1,3,4\}$

$D_{x_1} = \{10,50\}$

$D_{x_2} = \{10,20\}$

$D_{x_3} = \{40,50,80,90\}$

$D_{x_4} = \{40,50,70\}$

The reduced domains are:

$D_z = \{80,90\}$

$D_y = \{3\}$

$D_{x_1} = \{10,50\}$

$D_{x_2} = \{10,20\}$

$D_{x_3} = \{80,90\}$

$D_{x_4} = \{40,50,70\}$

Continuous relaxation of element

element$(y,(c_1,\ldots,c_n),z)$ is trivial.

The convex hull relaxation is $\min_i\{c_i\} \le z \le \max_i\{c_i\}$

element$(y,(x_1,\ldots,x_n),z)$ has the following relaxation

$$\sum_{i\in D_y}\frac{x_i}{m_i} - \left(\sum_{i\in D_y}\frac{1}{m_i}\right)z \ge -k+1$$

$$-\sum_{i\in D_y}\frac{x_i}{m_i} + \left(\sum_{i\in D_y}\frac{1}{m_i}\right)z \ge -k+1$$

provided $0 \le x_i \le m_i$ (and where $k = |D_y|$).

If $0 \le x_i \le m$ for all i, then the *convex hull* relaxation of element$(y, (x_1, \ldots, x_n), z)$ is

$$\sum_{j \in D_y} x_j - (k-1)m \le z \le \sum_{j \in D_y} x_j$$

plus bounds, where $k = |D_y|$.

Example...

$$\text{element}(y, (x_1, x_2), z)$$
$$0 \le x_1 \le 5$$
$$0 \le x_2 \le 5$$

The convex hull relaxation is:

$$x_1 + x_2 - 5 \le z \le x_1 + x_2$$
$$0 \le x_1 \le 5$$
$$0 \le x_2 \le 5$$

If $0 \le x_1 \le 4$ the above remains valid and we have

$$5x_1 + 4x_2 - 20 \le 9z \le 5x_1 + 4x_2 + 20$$

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:

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

Domain Reduction for All-different

$$\text{alldifferent}(y_1, \ldots y_n)$$

can be processed with an algorithm based on maximum cardinality bipartite matching and a theorem of Berge.

Domain Reduction for All-different

Consider the domains

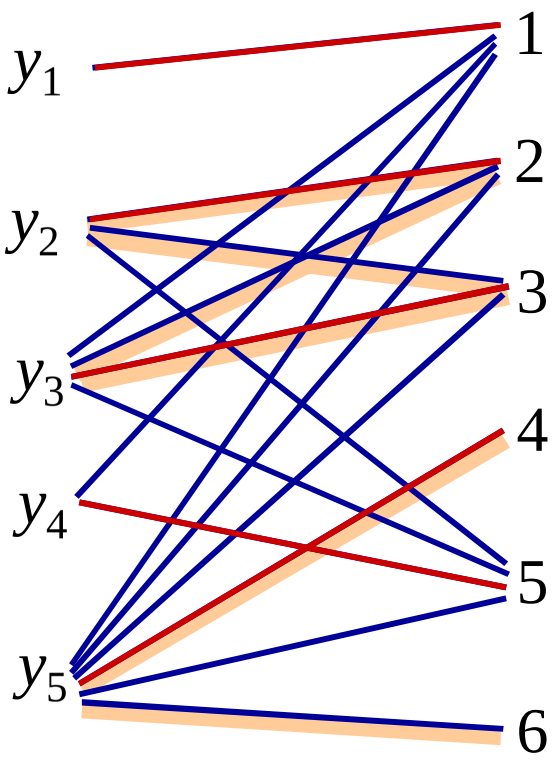
$$y_1 \in \{1\}$$

$$y_2 \in \{2,3,5\}$$

$$y_3 \in \{1,2,3,5\}$$

$$y_4 \in \{1,5\}$$

$$y_5 \in \{1,2,3,4,5,6\}$$



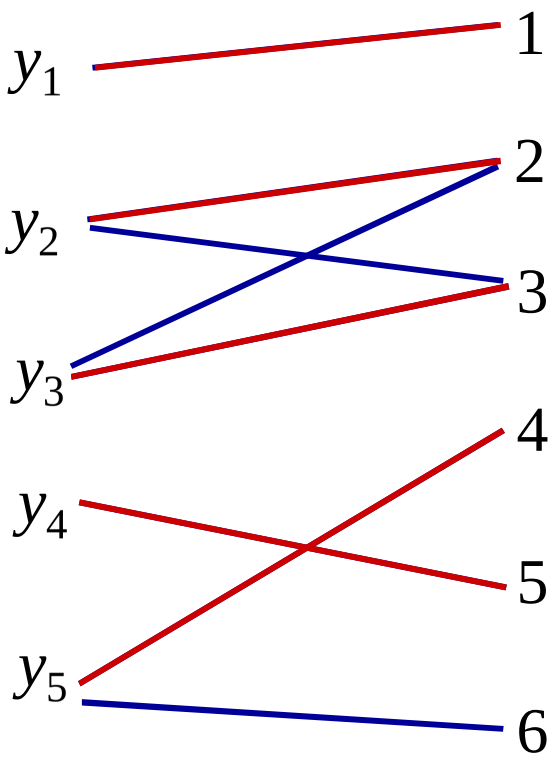
Indicate domains with edges

Find maximum cardinality bipartite matching.

Mark edges in alternating paths that start at an uncovered vertex.

Mark edges in alternating cycles.

Remove unmarked edges not in matching.



Indicate domains with edges

Find maximum cardinality bipartite matching.

Mark edges in alternating paths that start at an uncovered vertex.

Mark edges in alternating cycles.

Remove unmarked edges not in matching.

Domain Reduction for All-different

Domains have been reduced:

$y_1 \in \{1\}$ $\quad\quad$ $y_1 \in \{1\}$

$y_2 \in \{2,3,5\}$ $\quad\quad$ $y_2 \in \{2,3\}$

$y_3 \in \{1,2,3,5\}$ $\longrightarrow$ $y_3 \in \{2,3\}$

$y_4 \in \{1,5\}$ $\quad\quad$ $y_4 \in \{5\}$

$y_5 \in \{1,2,3,4,5,6\}$ $\quad\quad$ $y_5 \in \{4,6\}$

Domain Reduction for Cumulative

Use “edge finding” techniques.

Relaxation of cumulative

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

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

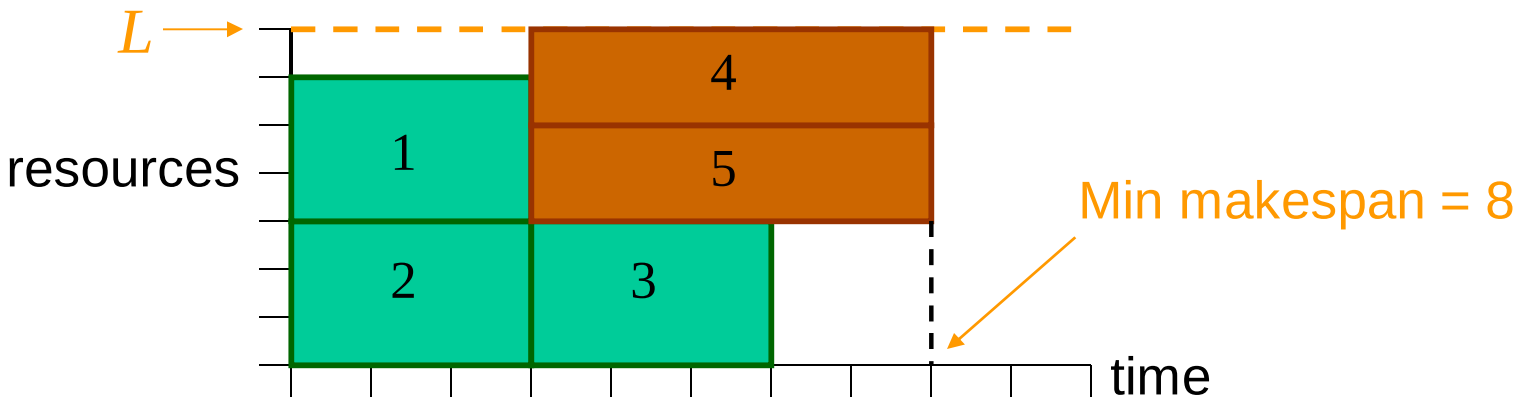
$$t_{j_1} + \cdots + t_{j_k} \geq \sum_{i=1}^{k} \left((k - i + \frac{1}{2}) \frac{r_i}{L} - \frac{1}{2} \right) d_i$$

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:

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

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.

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

$$x = \sum_k x^k$$

$$\sum_k y_k = 1$$

$$y_k \ge 0$$

Additional variables needed.

Can be extended to nonlinear systems.

"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 $0 \le x_j \le m_j$

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

where

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

Example:

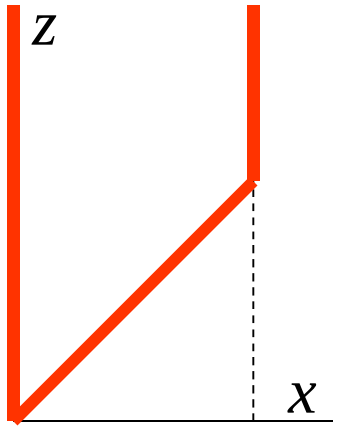
$$\begin{pmatrix} \text{no machine} \\ x = 0 \end{pmatrix} \vee \begin{pmatrix} \text{small machine} \\ z = 50 \\ x \le 5 \end{pmatrix} \vee \begin{pmatrix} \text{large machine} \\ z = 80 \\ x \le 10 \end{pmatrix}$$

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:

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