Combining Optimization and Constraint Programming

John Hooker
Carnegie Mellon University

June 2008

Optimization and Constraint Programming

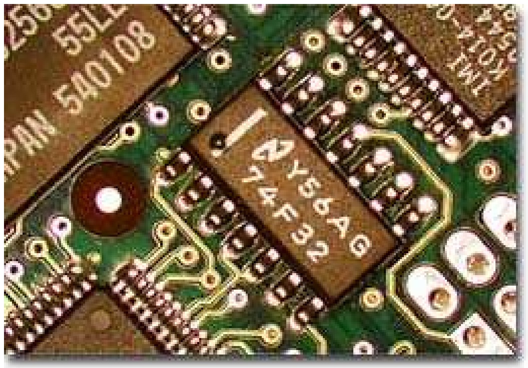
- Optimization: focus on mathematical programming.
  - 50+ years old
- Constraint programming
  - 20 years old
  - Developed in computer science/AI community

Optimization & Constraint programming

- Optimization methods rely heavily on **numerical calculation**.
  - Linear programming (LP)
  - Mixed integer/linear programming (MILP)
  - Nonlinear programming (NLP)
- Constraint programming relies heavily on **constraint propagation**
  - A form of logical inference

CP: Early commercial successes

- Circuit design (Siemens)



- Container port scheduling (Hong Kong and Singapore)

- Real-time control (Siemens, Xerox)

CP: Applications

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

CP: Applications

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

Why unify math programming and constraint programming?

- One-stop shopping**.**
  - One solver does it all.

Why unify math programming and constraint programming?

- One-stop shopping.
  - One solver does it all.
- Richer modeling framework.
  - Natural models, less debugging & development time.

Why unify math programming and constraint programming?

- One-stop shopping.
  - One solver does it all.
- Richer modeling framework.
  - Natural models, less debugging & development time.
- Computational speedup.
  - A selection of results…

Computational Advantage of Integrating MP and CP

Using CP + relaxation from MILP

	Problem	*Speedup*
Focacci, Lodi, Milano (1999)	Lesson timetabling	2 to 50 times faster than CP
Refalo (1999)	Piecewise linear costs	2 to 200 times faster than MILP
Hooker & Osorio (1999)	Flow shop scheduling, etc.	4 to 150 times faster than MILP.
Thorsteinsson & Ottosson (2001)	Product configuration	30 to 40 times faster than CP, MILP

Computational Advantage of Integrating MP and CP

Using CP + relaxation from MILP

	Problem	*Speedup*
Sellmann & Fahle (2001)	Automatic recording	1 to 10 times faster than CP, MILP
Van Hoeve (2001)	Stable set problem	Better than CP in less time
Bollapragada, Ghattas & Hooker (2001)	Structural design (nonlinear)	Up to 600 times faster than MILP. 2 problems: <6 min vs >20 hrs for MILP
Beck & Refalo (2003)	Scheduling with earliness & tardiness costs	Solved 67 of 90, CP solved only 12

Computational Advantage of Integrating MP and CP

Using CP-based Branch and Price

	Problem	*Speedup*
Yunes, Moura & de Souza (1999)	Urban transit crew scheduling	Optimal schedule for 210 trips, vs. 120 for traditional branch and price
Easton, Nemhauser & Trick (2002)	Traveling tournament scheduling	First to solve 8-team instance

Computational Advantage of Integrating MP and CP

Using CP/MILP Benders methods

	Problem	*Speedup*
Jain & Grossmann (2001)	Min-cost planning & scheduling	20 to 1000 times faster than CP, MILP
Thorsteinsson (2001)	Min-cost planning & scheduling	10 times faster than Jain & Grossmann
Timpe (2002)	Polypropylene batch scheduling at BASF	Solved previously insoluble problem in 10 min

Computational Advantage of Integrating MP and CP

Using CP/MILP Benders methods

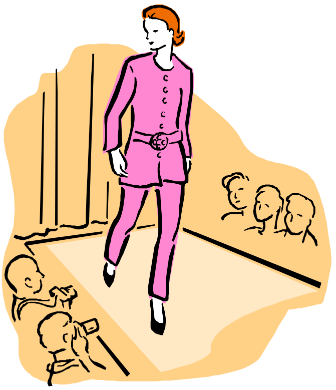
	Problem	*Speedup*
Benoist, Gaudin, Rottembourg (2002)	Call center scheduling	Solved twice as many instances as traditional Benders
Hooker (2004)	Min-cost, min-makespan planning & cumulative scheduling	100-1000 times faster than CP, MILP
Hooker (2005)	Min tardiness planning & cumulative scheduling	10-1000 times faster than CP, MILP

An Exercise in Synthesis

- **Analysis** takes things apart.
- **Synthesis** looks for commonality.

- I will provide an **overview** of several examples.
  - Look for **common patterns**, not details.

Modeling is key



- In math programming, the model describes the problem but doesn't suggest how to solve it.
- In CP, each constraint invokes a procedure that screens out solutions unacceptable that that constraint.
- This can be extended to a unified framework.
  - Model consists of **metaconstraints**.
  - Each metaconstraint "knows" how to combine MP and CP to exploit its structure.

The basic algorithm

- **Search:** Enumerate problem restrictions
  - Tree search (branching)
  - Constraint-based (nogood-based) search

The basic algorithm

- **Search:** Enumerate problem restrictions
  - Tree search (branching)
  - Constraint-based (nogood-based) search
- **Infer:** Deduce constraints from current restriction

The basic algorithm

- **Search:** Enumerate problem restrictions
  - Tree search (branching)
  - Constraint-based (nogood-based) search
- **Infer:** Deduce constraints from current restriction
- **Relax**: Solve relaxation of current restriction

Unifying framework

- **Existing methods** are special cases of this framework**.**
- **Integrated methods** are also special cases.
  - Select an overall **search** scheme.
  - Select **inference** methods as needed from CP, OR.
  - Select **relaxation** methods as needed.

Some existing methods – Branching

- Constraint solvers (CP)
  - **Search:** Branching on domains
  - **Inference:** Constraint propagation, filtering
  - **Relaxation:** Domain store
- Mixed integer programming (OR)
  - **Search:** Branch and bound
  - **Inference:** Cutting planes
  - **Relaxation:** Linear programming

Some existing methods – Constraint-based search

- SAT solvers (CP)
  - **Search:** Branching on variables
  - **Inference:** Unit clause rule, clause learning (nogoods)
  - **Relaxation:** Conflict clauses
- Benders decomposition (OR)
  - **Search:** Enumeration of subproblems
  - **Inference:** Benders cuts (nogoods)
  - **Relaxation:** Master problem

Outline

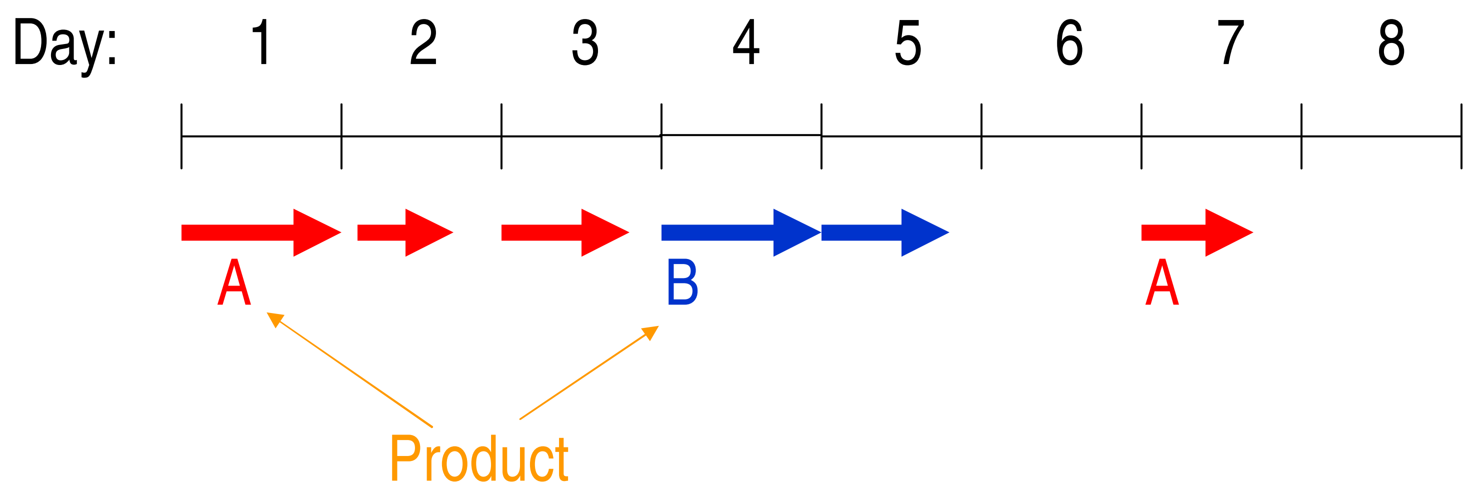
Example problems to illustrate integrated approach

- Simpler modeling
  - *Lot sizing and scheduling*
- Branching search
  - *Freight transfer* – illustration of the algorithm
  - *Product configuration* – easier modeling & faster solution
  - *Airline crew scheduling* – CP-based branch and price
- Constraint-based search
  - *Machine scheduling* – Logic-based Benders algorithm
  - *Success stories from BASF, Barbot, Peugeot-Citroën*
- Software

Example: Lot sizing and scheduling

Simplified modeling

Lot sizing and scheduling



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

Integer programming model

(Wolsey)

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

Many variables

Integrated 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

Production capacity

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

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

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

Integrated model

Minimize holding and setup costs

Variable indices

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

Inventory balance

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

Production capacity

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

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

Production level of product *i* in period *t*

Product manufactured in period *t*

Example: Freight Transfer

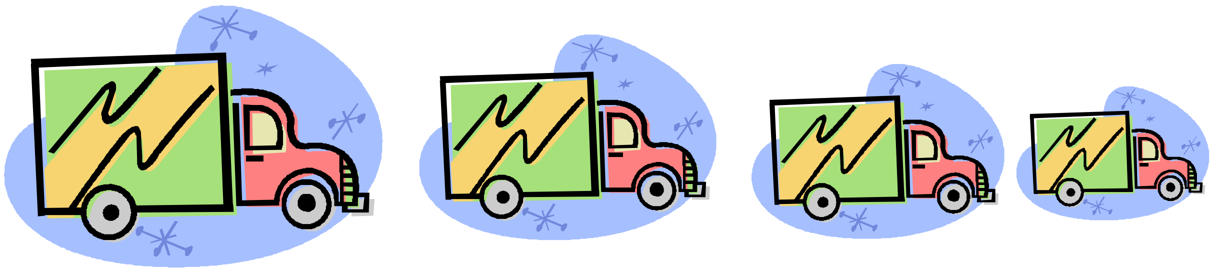
Branch-and-bound search with interval propagation and cutting planes

This example illustrates:

- **Branching** on variables, with pruning based on bounds.
- **Propagation** based on:
  - Interval propagation.
  - Cutting planes (**knapsack cuts**).
- **Relaxation** based on linear programming.

Freight Transfer

- Transport 42 tons of freight using 8 trucks, which come in 4 sizes…



Truck size	Number available	Capacity (tons)	Cost per truck
1	3	7	90
2	3	5	60
3	3	4	50
4	3	3	40

Number of trucks of type 1

$$\min\ 90x_1 + 60x_2 + 50x_3 + 40x_4$$

$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$

$$x_1 + x_2 + x_3 + x_4 \leq 8$$

$$x_i \in \{0,1,2,3\}$$

Truck type	Number available	Capacity (tons)	Cost per truck
1	3	7	90
2	3	5	60
3	3	4	50
4	3	3	40

Number of trucks of type 1

$$\min\ 90x_1 + 60x_2 + 50x_3 + 40x_4$$
$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$
$$x_1 + x_2 + x_3 + x_4 \leq 8$$
$$x_i \in \{0,1,2,3\}$$

Knapsack metaconstraint "knows" which inference and relaxation techniques to use.

Truck type	Number available	Capacity (tons)	Cost per truck
1	3	7	90
2	3	5	60
3	3	4	50
4	3	3	40

Number of trucks of type 1

$$\min\ 90x_1 + 60x_2 + 50x_3 + 40x_4$$
$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$
$$x_1 + x_2 + x_3 + x_4 \leq 8$$
$$\boxed{x_i \in \{0,1,2,3\}}$$

Domain metaconstraint "knows" how to branch

Truck type	Number available	Capacity (tons)	Cost per truck
1	3	7	90
2	3	5	60
3	3	4	50
4	3	3	40

Bounds propagation

$$\min\ 90x_1 + 60x_2 + 50x_3 + 40x_4$$
$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$
$$x_1 + x_2 + x_3 + x_4 \leq 8$$
$$x_i \in \{0,1,2,3\}$$

$$x_1 \geq \left\lceil \frac{42 - 5\cdot 3 - 4\cdot 3 - 3\cdot 3}{7} \right\rceil = 1$$

Bounds propagation

$$\min\ 90x_1 + 60x_2 + 50x_3 + 40x_4$$
$$7x_1 + 5x_2 + 4x_3 + 3x_4 \ge 42$$
$$x_1 + x_2 + x_3 + x_4 \le 8$$
$$x_1 \in \{1,2,3\}, \quad x_2, x_3, x_4 \in \{0,1,2,3\}$$

Reduced domain

$$x_1 \ge \left\lceil \frac{42 - 5\cdot 3 - 4\cdot 3 - 3\cdot 3}{7} \right\rceil = 1$$

Continuous relaxation

$$\min\ 90x_1 + 60x_2 + 50x_3 + 40x_4$$
$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$
$$x_1 + x_2 + x_3 + x_4 \leq 8$$
$$0 \leq x_i \leq 3, \quad x_1 \geq 1$$

Replace domains with bounds

This is a **linear programming problem**, which is easy to solve.

Its optimal value provides a lower bound on optimal value of original problem.

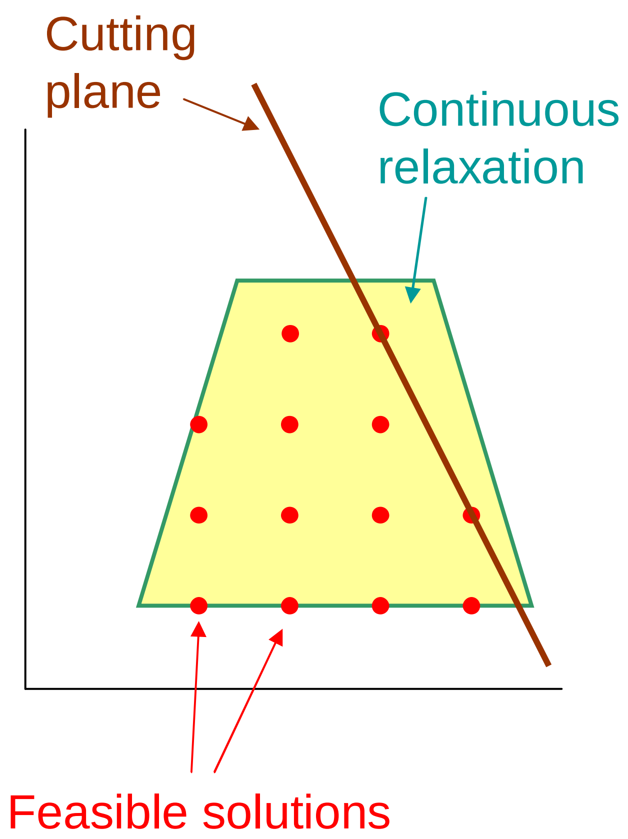
Cutting planes (valid inequalities)

$$\min\ 90x_1 + 60x_2 + 50x_3 + 40x_4$$
$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$
$$x_1 + x_2 + x_3 + x_4 \leq 8$$
$$0 \leq x_i \leq 3, \quad x_1 \geq 1$$

We can create a **tighter** relaxation (larger minimum value) with the addition of **cutting planes**.

Cutting planes (valid inequalities)

$$\min\ 90x_1 + 60x_2 + 50x_3 + 40x_4$$
$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$
$$x_1 + x_2 + x_3 + x_4 \leq 8$$
$$0 \leq x_i \leq 3, \quad x_1 \geq 1$$



All feasible solutions of the original problem satisfy a cutting plane (i.e., it is **valid**).

But a cutting plane may exclude ("**cut off**") solutions of the continuous relaxation.

Cutting planes (valid inequalities)

$$\min\ 90x_1 + 60x_2 + 50x_3 + 40x_4$$
$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$
$$x_1 + x_2 + x_3 + x_4 \leq 8$$
$$0 \leq x_i \leq 3, \quad x_1 \geq 1$$

{1,2} is a **packing**

…because $7x_1 + 5x_2$ alone cannot satisfy the inequality, even with $x_1 = x_2 = 3$.

Cutting planes (valid inequalities)

$$\min\ 90x_1 + 60x_2 + 50x_3 + 40x_4$$

$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$

$$x_1 + x_2 + x_3 + x_4 \leq 8$$

$$0 \leq x_i \leq 3, \quad x_1 \geq 1$$

{1,2} is a **packing**

So, $4x_3 + 3x_4 \geq 42 - (7 \cdot 3 + 5 \cdot 3)$

Cutting planes (valid inequalities)

$$\min\ 90x_1 + 60x_2 + 50x_3 + 40x_4$$
$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$
$$x_1 + x_2 + x_3 + x_4 \leq 8$$
$$0 \leq x_i \leq 3,\quad x_1 \geq 1$$

{1,2} is a **packing**

So, $4x_3 + 3x_4 \geq 42 - (7 \cdot 3 + 5 \cdot 3)$

Knapsack cut

which implies

$$x_3 + x_4 \geq \left\lceil \frac{42 - (7 \cdot 3 + 5 \cdot 3)}{\max\{4,3\}} \right\rceil = 2$$

Cutting planes (valid inequalities)

$$\min\ 90x_1 + 60x_2 + 50x_3 + 40x_4$$
$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$
$$x_1 + x_2 + x_3 + x_4 \leq 8$$
$$0 \leq x_i \leq 3, \quad x_1 \geq 1$$

Maximal Packings	**Knapsack cuts**
{1,2}	$x_3 + x_4 \geq 2$
{1,3}	$x_2 + x_4 \geq 2$
{1,4}	$x_2 + x_3 \geq 3$

Knapsack cuts corresponding to nonmaximal packings can be nonredundant.

Continuous relaxation with cuts

$$\min\ 90x_1 + 60x_2 + 50x_3 + 40x_4$$
$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$
$$x_1 + x_2 + x_3 + x_4 \leq 8$$
$$0 \leq x_i \leq 3, \quad x_1 \geq 1$$
$$\boxed{\begin{aligned} x_3 + x_4 &\geq 2 \\ x_2 + x_4 &\geq 2 \\ x_2 + x_3 &\geq 3 \end{aligned}}$$

Knapsack cuts

Optimal value of 523.3 is a lower bound on optimal value of original problem.

Branch-infer-and-relax tree

$x_1 \in \{\ \ 123\}$
$x_2 \in \{0123\}$
$x_3 \in \{0123\}$
$x_4 \in \{0123\}$
$x = (2⅓,3,2⅔,0)$
value = 523⅓

Propagate bounds and solve relaxation of original problem.

Branch-infer-and-relax tree

Branch on a variable with nonintegral value in the relaxation.

$x_1 \in \{\ \ 123\}$
$x_2 \in \{0123\}$
$x_3 \in \{0123\}$
$x_4 \in \{0123\}$
$x = (2⅓,3,2⅔,0)$
value = 523⅓

$x_1 \in \{1,2\}$ $\qquad$ $x_1 = 3$

Branch-infer-and-relax tree

Propagate bounds and solve relaxation.

Since relaxation is infeasible, backtrack.

$x_1 \in \{\ \ 123\}$
$x_2 \in \{0123\}$
$x_3 \in \{0123\}$
$x_4 \in \{0123\}$
$x = (2⅓,3,2⅔,0)$
value = 523⅓

$x_1 \in \{1,2\}$ $\qquad$ $x_1 = 3$

$x_1 \in \{\ 12\ \}$
$x_2 \in \{\ \ \ 23\}$
$x_3 \in \{\ 123\}$
$x_4 \in \{\ 123\}$
infeasible relaxation

Branch-infer-and-relax tree

Propagate bounds and solve relaxation.

Branch on nonintegral variable.

$x_1 \in \{\ \ 123\}$
$x_2 \in \{0123\}$
$x_3 \in \{0123\}$
$x_4 \in \{0123\}$
$x = (2⅓,3,2⅔,0)$
value = 523⅓

$x_1 \in \{1,2\}$

$x_1 \in \{\ \ 12\ \ \}$
$x_2 \in \{\ \ \ 23\}$
$x_3 \in \{\ \ 123\}$
$x_4 \in \{\ \ 123\}$
infeasible relaxation

$x_1 = 3$

$x_1 \in \{\ \ \ \ \ 3\}$
$x_2 \in \{0123\}$
$x_3 \in \{0123\}$
$x_4 \in \{0123\}$
$x = (3,2.6,2,0)$
value = 526

$x_2 \in \{0,1,2\}$

$x_2 = 3$

Branch-infer-and-relax tree

Branch again.

$x_1 \in \{\ \ 123\}$
$x_2 \in \{0123\}$
$x_3 \in \{0123\}$
$x_4 \in \{0123\}$
$x = (2⅓,3,2⅔,0)$
value = 523⅓

$x_1 \in \{1,2\}$

$x_1 \in \{\ 12\ \ \}$
$x_2 \in \{\ \ \ 23\}$
$x_3 \in \{\ 123\}$
$x_4 \in \{\ 123\}$
infeasible relaxation

$x_1 = 3$

$x_1 \in \{\ \ \ \ \ 3\}$
$x_2 \in \{0123\}$
$x_3 \in \{0123\}$
$x_4 \in \{0123\}$
$x = (3,2.6,2,0)$
value = 526

$x_2 \in \{0,1,2\}$

$x_1 \in \{\ \ \ \ \ 3\}$
$x_2 \in \{012\ \ \}$
$x_3 \in \{\ 123\}$
$x_4 \in \{0123\}$
$x = (3,2,2¾,0)$
value = 527½

$x_3 \in \{1,2\}$

$x_3 = 3$

$x_2 = 3$

Branch-infer-and-relax tree

$x_1 \in \{\ \ 123\}$
$x_2 \in \{0123\}$
$x_3 \in \{0123\}$
$x_4 \in \{0123\}$
$x = (2⅓,3,2⅔,0)$
value = 523⅓

$x_1 \in \{1,2\}$

$x_1 \in \{\ \ 12\ \ \}$
$x_2 \in \{\ \ \ \ 23\}$
$x_3 \in \{\ \ 123\}$
$x_4 \in \{\ \ 123\}$
infeasible relaxation

$x_1 = 3$

$x_1 \in \{\ \ \ \ \ \ 3\}$
$x_2 \in \{0123\}$
$x_3 \in \{0123\}$
$x_4 \in \{0123\}$
$x = (3,2.6,2,0)$
value = 526

$x_2 \in \{0,1,2\}$

$x_1 \in \{\ \ \ \ \ \ 3\}$
$x_2 \in \{012\ \ \}$
$x_3 \in \{\ \ 123\}$
$x_4 \in \{0123\}$
$x = (3,2,2¾,0)$
value = 527½

$x_2 = 3$

$x_3 \in \{1,2\}$

$x_1 \in \{\ \ \ \ \ \ 3\}$
$x_2 \in \{\ \ 12\ \ \}$
$x_3 \in \{\ \ 12\ \ \}$
$x_4 \in \{\ \ 123\}$
$x = (3,2,2,1)$
value = 530
feasible solution

$x_3 = 3$

Solution of relaxation is integral and therefore feasible in the original problem.

This becomes the **incumbent solution**.

Branch-infer-and-relax tree

Solution is nonintegral, but we can backtrack because value of relaxation is no better than incumbent solution.

$x_1 \in \{\ 123\}$
$x_2 \in \{0123\}$
$x_3 \in \{0123\}$
$x_4 \in \{0123\}$
$x = (2\tfrac{1}{3},3,2\tfrac{2}{3},0)$
value = $523\tfrac{1}{3}$

$x_1 \in \{1,2\}$

$x_1 \in \{\ 12\ \}$
$x_2 \in \{\ \ 23\}$
$x_3 \in \{\ 123\}$
$x_4 \in \{\ 123\}$
infeasible relaxation

$x_1 = 3$

$x_1 \in \{\ \ \ 3\}$
$x_2 \in \{0123\}$
$x_3 \in \{0123\}$
$x_4 \in \{0123\}$
$x = (3,2.6,2,0)$
value = 526

$x_2 \in \{0,1,2\}$

$x_1 \in \{\ \ \ 3\}$
$x_2 \in \{012\ \}$
$x_3 \in \{\ 123\}$
$x_4 \in \{0123\}$
$x = (3,2,2\tfrac{3}{4},0)$
value = $527\tfrac{1}{2}$

$x_2 = 3$

$x_3 \in \{1,2\}$

$x_1 \in \{\ \ \ 3\}$
$x_2 \in \{\ 12\ \}$
$x_3 \in \{\ 12\ \}$
$x_4 \in \{\ 123\}$
$x = (3,2,2,1)$
value = 530
feasible solution

$x_3 = 3$

$x_1 \in \{\ \ \ 3\}$
$x_2 \in \{012\ \}$
$x_3 \in \{\ \ \ 3\}$
$x_4 \in \{012\ \}$
$x = (3,1\tfrac{1}{2},3,\tfrac{1}{2})$
value = 530
backtrack due to bound

Branch-infer-and-relax tree

Another feasible solution found.

No better than incumbent solution, which is optimal because search has finished.

$x_1 \in \{\ \ 123\}$
$x_2 \in \{0123\}$
$x_3 \in \{0123\}$
$x_4 \in \{0123\}$
$x = (2⅓,3,2⅔,0)$
value = 523⅓

Branch $x_1 \in \{1,2\}$:

$x_1 \in \{\ 12\ \}$
$x_2 \in \{\ \ 23\}$
$x_3 \in \{\ 123\}$
$x_4 \in \{\ 123\}$
infeasible relaxation

Branch $x_1 = 3$:

$x_1 \in \{\ \ \ 3\}$
$x_2 \in \{0123\}$
$x_3 \in \{0123\}$
$x_4 \in \{0123\}$
$x = (3,2.6,2,0)$
value = 526

Branch $x_2 \in \{0,1,2\}$:

$x_1 \in \{\ \ \ 3\}$
$x_2 \in \{012\ \}$
$x_3 \in \{\ 123\}$
$x_4 \in \{0123\}$
$x = (3,2,2¾,0)$
value = 527½

Branch $x_3 \in \{1,2\}$:

$x_1 \in \{\ \ \ 3\}$
$x_2 \in \{\ 12\ \}$
$x_3 \in \{\ 12\ \}$
$x_4 \in \{\ 123\}$
$x = (3,2,2,1)$
value = 530
feasible solution

Branch $x_3 = 3$:

$x_1 \in \{\ \ \ 3\}$
$x_2 \in \{012\ \}$
$x_3 \in \{\ \ \ 3\}$
$x_4 \in \{012\ \}$
$x = (3,1½,3,½)$
value = 530
backtrack due to bound

Branch $x_2 = 3$:

$x_1 \in \{\ \ \ 3\}$
$x_2 \in \{\ \ \ 3\}$
$x_3 \in \{012\ \}$
$x_4 \in \{012\ \}$
$x = (3,3,0,2)$
value = 530
feasible solution

Branch-infer-and-relax tree

Two optimal solutions found.

In general, not all optimal solutions are found,

$x_1 \in \{123\}$
$x_2 \in \{0123\}$
$x_3 \in \{0123\}$
$x_4 \in \{0123\}$
$x = (2⅓,3,2⅔,0)$
value = 523⅓

$x_1 \in \{1,2\}$

$x_1 \in \{12\}$
$x_2 \in \{23\}$
$x_3 \in \{123\}$
$x_4 \in \{123\}$
infeasible relaxation

$x_1 = 3$

$x_1 \in \{3\}$
$x_2 \in \{0123\}$
$x_3 \in \{0123\}$
$x_4 \in \{0123\}$
$x = (3,2.6,2,0)$
value = 526

$x_2 \in \{0,1,2\}$

$x_1 \in \{3\}$
$x_2 \in \{012\}$
$x_3 \in \{123\}$
$x_4 \in \{0123\}$
$x = (3,2,2¾,0)$
value = 527½

$x_2 = 3$

$x_1 \in \{3\}$
$x_2 \in \{3\}$
$x_3 \in \{012\}$
$x_4 \in \{012\}$
$x = (3,3,0,2)$
value = 530
optimal solution

$x_3 \in \{1,2\}$

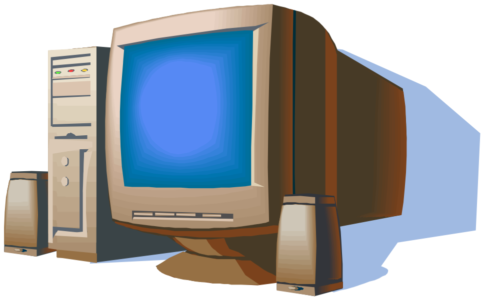
$x_1 \in \{3\}$
$x_2 \in \{12\}$
$x_3 \in \{12\}$
$x_4 \in \{123\}$
$x = (3,2,2,1)$
value = 530
optimal solution

$x_3 = 3$

$x_1 \in \{3\}$
$x_2 \in \{012\}$
$x_3 \in \{3\}$
$x_4 \in \{012\}$
$x = (3,1½,3,½)$
value = 530
backtrack due to bound

Other types of cutting planes

- Lifted 0-1 knapsack inequalities
- Clique inequalities
- Gomory cuts
- Mixed integer rounding cuts
- Disjunctive cuts
- Specialized cuts
  - Flow cuts (fixed charge network flow problem)
  - Comb inequalities (traveling salesman problem)
  - Many, many others

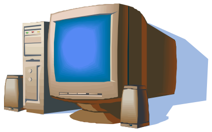


Example: Product Configuration

Branch-and-bound search with propagation and relaxation of **variable indices**.

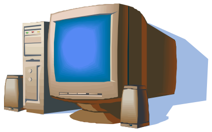
From: Thorsteinsson and Ottosson (2001)

- **This example illustrates:**



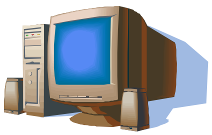
- **Propagation** of variable indices.
  - **Variable index** is converted to a specially structured *element* constraint.
  - Specially structured **filtering** for *element.*
  - Valid **knapsack** cuts are derived and propagated.

- **This example illustrates:**

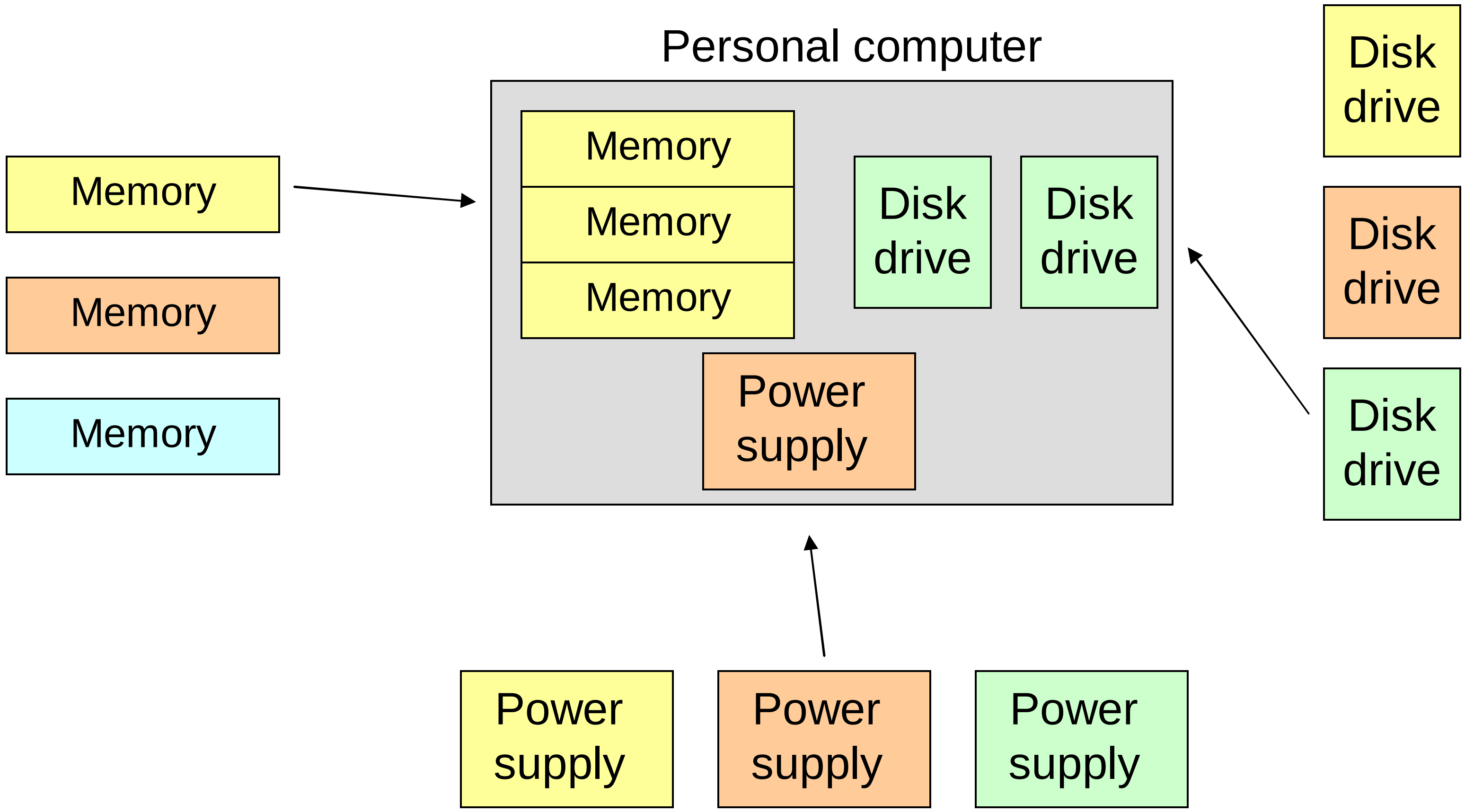


- **Propagation** of variable indices.
  - **Variable index** is converted to a specially structured *element* constraint.
  - Specially structured **filtering** for *element.*
  - Valid **knapsack** cuts are derived and propagated.
- **Relaxation** of variable indices.
  - Element is interpreted as a **disjunction** of linear systems.
  - **Convex hull** relaxation for disjunction is used.

The problem



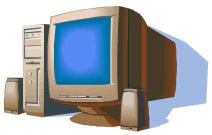
Choose what type of each component, and how many



Problem data

Component i	Type k	Net power generation A_{1jk}	Disk space A_{2jk}	Memory capacity A_{3jk}	Weight A_{4jk}	Max number used
1. Power supply	A	70	0	0	200	1
	B	100	0	0	250	
	C	150	0	0	350	
2. Disk drive	A	-30	500	0	140	3
	B	-50	800	0	300	
3. Memory chip	A	-20	0	250	20	3
	B	-25	0	300	25	
	C	-30	0	400	30	
Lower bound L_j		0	700	850	0	
Upper bound U_j		∞	∞	∞	∞	
Unit cost c_j		0	0	0	1	

Model of the problem



Unit cost of producing attribute *j*

Amount of attribute *j* produced by type t_i of component *i*

Amount of attribute *j* produced (< 0 if consumed): *memory, heat, power, weight, etc.*

$$\min \sum_j c_j v_j$$

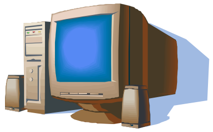
$$v_j = \sum_{ik} q_i A_{ijt_i}, \text{ all } j$$

$$L_j \le v_j \le U_j, \text{ all } j$$

t_i is a **variable index**

Quantity of component *i* installed

Model of the problem



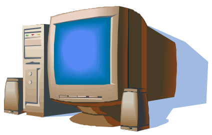
Linear inequality metaconstraint

$$\boxed{\min \sum_j c_j v_j}$$

$$v_j = \sum_{ik} q_i A_{ijt_i}, \text{ all } j$$

$$\boxed{L_j \le v_j \le U_j, \text{ all } j}$$

Model of the problem



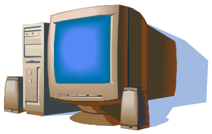
$$\min \sum_j c_j v_j$$

Indexed linear metaconstraint

$$\boxed{v_j = \sum_{ik} q_i A_{ijt_i}, \text{ all } j}$$

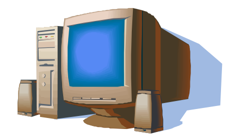
$$L_j \le v_j \le U_j, \text{ all } j$$

To solve it:



- **Branch** on domains of t_i and q_i.
- **Propagate** *element* constraints and bounds on v_j.
  - Derive and propagate **knapsack** cuts.
- **Relax** *element*.
  - **Convex hull** relaxation for disjunction.

Propagation



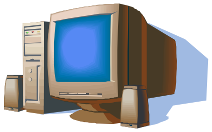
$$\min \sum_j c_j v_j$$

$$v_j = \sum_{ik} q_i A_{ijt_i}, \text{ all } j$$

$$\boxed{L_j \le v_j \le U_j, \text{ all } j}$$

This is propagated in the usual way

Propagation



$$v_j = \sum_i z_i, \text{ all } j$$

$$\text{element}\left(t_i, (q_i, A_{ij1}, \ldots, q_i A_{ijn}), z_i\right), \text{ all } i, j$$

$$\min \sum_j c_j v_j$$

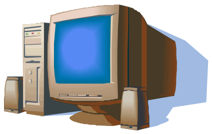
$$\boxed{v_j = \sum_{ik} q_i A_{ijt_i}, \text{ all } j}$$

$$\boxed{L_j \le v_j \le U_j, \text{ all } j}$$

This is rewritten as

This is propagated in the usual way

Propagation



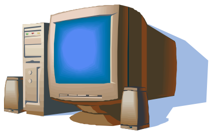
$$v_j = \sum_i z_i, \text{ all } j$$

$$\text{element}\left(t_i, (q_i, A_{ij1}, \ldots, q_i A_{ijn}), z_i\right), \text{ all } i, j$$

This can be propagated by

(a) using specialized **filters** for *element* constraints of this form…

Propagation



$$v_j = \sum_i z_i, \text{ all } j$$

$$\text{element}\left(t_i, (q_i, A_{ij1}, \ldots, q_i A_{ijn}), z_i\right), \text{ all } i, j$$

This is propagated by
(a) using specialized **filters** for *element* constraints of this form,
(b) adding **knapsack cuts** for the valid inequalities:

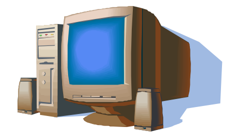
$$\sum_i \max_{k \in D_{t_i}} \left\{A_{ijk}\right\} q_i \ge \underline{v}_j, \text{ all } j$$

$$\sum_i \min_{k \in D_{t_i}} \left\{A_{ijk}\right\} q_i \le \overline{v}_j, \text{ all } j$$

and (c) propagating the knapsack cuts.

$[\underline{v}_j, \overline{v}_j]$ is current domain of v_j

Relaxation



$$\min \sum_j c_j v_j$$

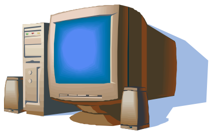
$$v_j = \sum_{ik} q_i A_{ijt_i}, \text{ all } j$$

$$\boxed{L_j \le v_j \le U_j, \text{ all } j}$$

This is relaxed as

$$\underline{v}_j \le v_j \le \bar{v}_j$$

Relaxation



$$v_j = \sum_i z_i, \text{ all } j$$

$$\text{element}\left(t_i, (q_i A_{ij1}, \ldots, q_i A_{ijn}), z_i\right), \text{ all } i, j$$

$$\min \sum_j c_j v_j$$

$$\boxed{v_j = \sum_{ik} q_i A_{ijt_i}, \text{ all } j}$$

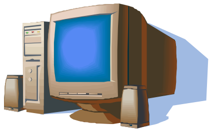
$$\boxed{L_j \le v_j \le U_j, \text{ all } j}$$

This is relaxed by relaxing *this* and adding the knapsack cuts.

This is relaxed as

$$\underline{v}_j \le v_j \le \overline{v}_j$$

Relaxation



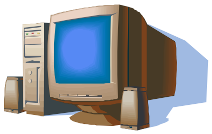
$$v_j = \sum_i z_i, \text{ all } j$$

$$\text{element}\left(t_i, (q_i A_{ij1}, \ldots, q_i A_{ijn}), z_i\right), \text{ all } i, j$$

This is relaxed by writing each *element* constraint as a **disjunction** of linear systems and writing a **convex hull** relaxation of the disjunction:

$$z_i = \sum_{k \in D_{t_i}} A_{ijk} q_{ik}, \quad q_i = \sum_{k \in D_{t_i}} q_{ik}$$

Relaxation



So the following LP relaxation is solved at each node of the search tree to obtain a lower bound:

$$\min \sum_j c_j v_j$$

$$v_j = \sum_i \sum_{k \in D_{t_i}} A_{ijk} q_{ik}, \text{ all } j$$

$$q_j = \sum_{k \in D_{t_i}} q_{ik}, \text{ all } i$$

$$\underline{v}_j \le v_j \le \overline{v}_j, \text{ all } j$$

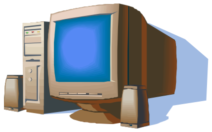
$$\underline{q}_i \le q_i \le \overline{q}_i, \text{ all } i$$

$$\text{knapsack cuts for } \sum_i \max_{k \in D_{t_i}} \{A_{ijk}\} q_i \ge \underline{v}_j, \text{ all } j$$

$$\text{knapsack cuts for } \sum_i \min_{k \in D_{t_i}} \{A_{ijk}\} q_i \le \overline{v}_j, \text{ all } j$$

$$q_{ik} \ge 0, \text{ all } i, k$$

Solution of the example



After propagation, the solution of the relaxation is feasible at the root node. No branching needed.

$$\min \sum_j c_j v_j$$

$$v_j = \sum_i \sum_{k \in D_{t_i}} A_{ijk} q_{ik}, \text{ all } j$$

$$q_j = \sum_{k \in D_{t_i}} q_{ik}, \text{ all } i$$

$$\underline{v}_j \le v_j \le \bar{v}_j, \text{ all } j$$

$$\underline{q}_i \le q_i \le \bar{q}_i, \text{ all } i$$

$$\text{knapsack cuts for } \sum_i \max_{k \in D_{t_i}} \{A_{ijk}\} q_i \ge \underline{v}_j, \text{ all } j$$

$$\text{knapsack cuts for } \sum_i \min_{k \in D_{t_i}} \{A_{ijk}\} q_i \le \bar{v}_j, \text{ all } j$$

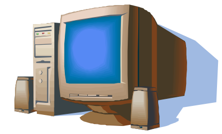
$$q_{ik} \ge 0, \text{ all } i,k$$

$$q_1, q_{1C} = 1 \rightarrow t_1 = \text{C}$$

$$q_2, q_{2A} = 2 \rightarrow t_2 = \text{A}$$

$$q_3, q_{3B} = 3 \rightarrow t_3 = \text{B}$$

Solution of the example



After propagation, the solution of the relaxation is feasible at the root node. No branching needed.

$$
\begin{aligned}
&\min \ \sum_j c_j v_j \\
&v_j = \sum_i \sum_{k \in D_{t_i}} A_{ijk} q_{ik}, \ \text{all } j \\
&q_j = \sum_{k \in D_{t_i}} q_{ik}, \ \text{all } i \\
&\underline{v}_j \le v_j \le \overline{v}_j, \ \text{all } j \\
&\underline{q}_i \le q_i \le \overline{q}_i, \ \text{all } i \\
&\text{knapsack cuts for } \sum_i \max_{k \in D_{t_i}} \{A_{ijk}\} q_i \ge \underline{v}_j, \ \text{all } j \\
&\text{knapsack cuts for } \sum_i \min_{k \in D_{t_i}} \{A_{ijk}\} q_i \le \overline{v}_j, \ \text{all } j \\
&q_{ik} \ge 0, \ \text{all } i, k
\end{aligned}
$$

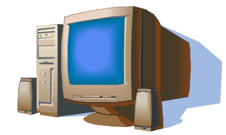
$$\boxed{q_1,\, q_{1C} = 1} \rightarrow \ t_1 = \mathrm{C}$$

$$q_2,\, q_{2A} = 2 \rightarrow \ t_2 = \mathrm{A}$$

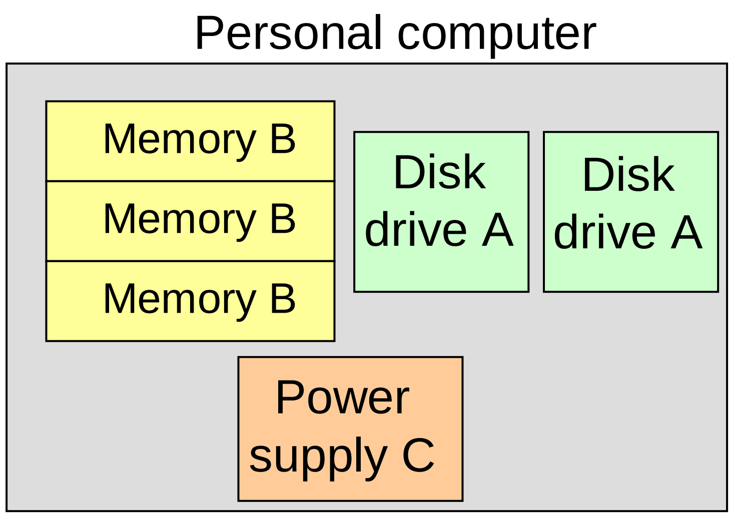
$$q_3,\, q_{3B} = 3 \rightarrow \ t_3 = \mathrm{B}$$

q_1 is integral,
only one q_{1k} is positive

Solution of the example



After propagation, the solution of the relaxation is feasible at the root node. No branching needed.

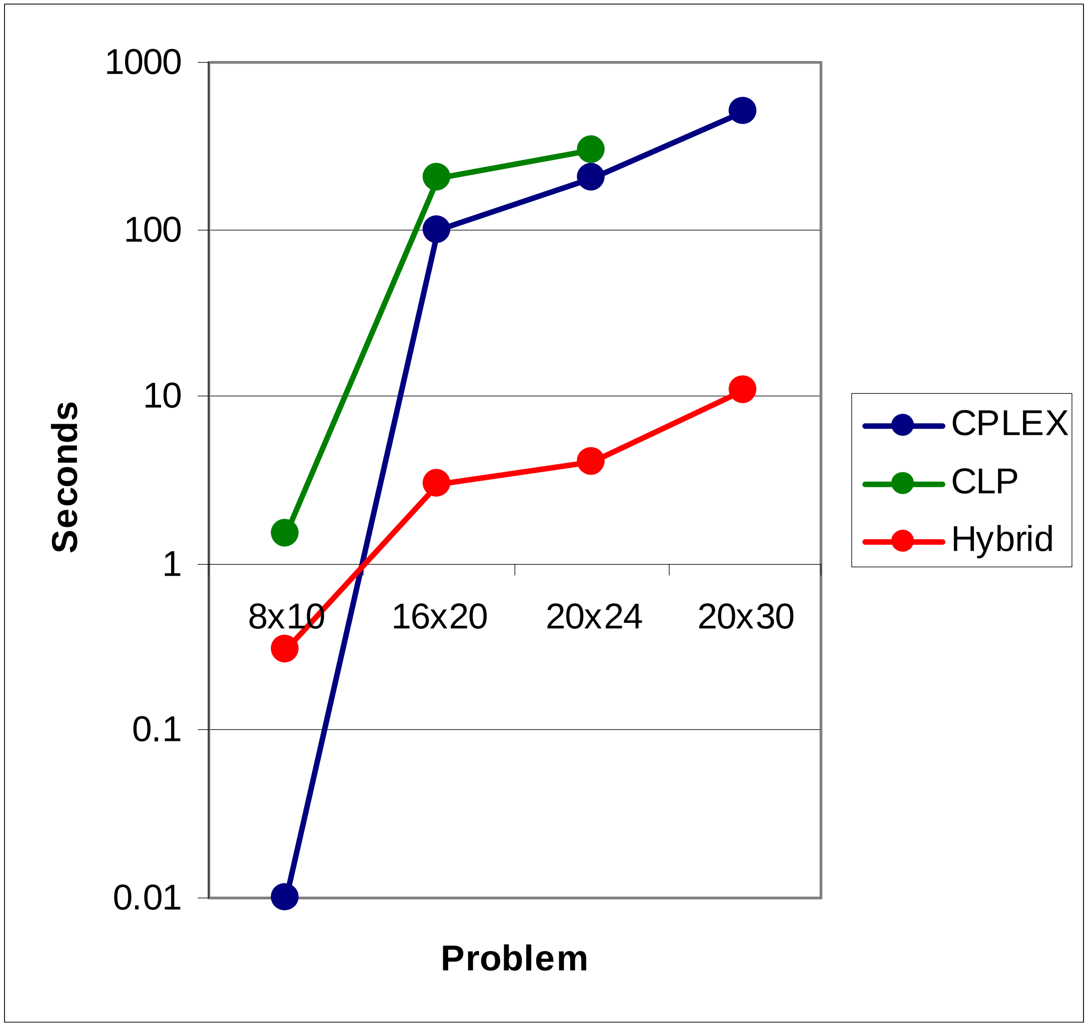


$$q_1,\ q_{1C} = 1 \rightarrow\ t_1 = \mathrm{C}$$

$$q_2,\ q_{2A} = 2 \rightarrow\ t_2 = \mathrm{A}$$

$$q_3,\ q_{3B} = 3 \rightarrow\ t_3 = \mathrm{B}$$

Computational Results



Example: Airline Crew Scheduling

Branch and price in which a linear relaxation of an MILP is solved by CP-based column generation

From: Fahle et al. (2002)

This example illustrates:

- Overall **mixed integer programming** framework.
- **Linear relaxation** solved by CP-based column generation.

Solving an LP by column generation

Suppose the LP relaxation of an integer programming problem has a huge number of variables:

$$\begin{aligned} &\min \ cx \\ &Ax = b \\ &x \geq 0 \end{aligned}$$

Solving an LP by column generation

Suppose the LP relaxation of an integer programming problem has a huge number of variables:

$$\min \; cx$$
$$Ax = b$$
$$x \geq 0$$

We will solve a **restricted master problem**, which has a small subset of the variables:

$$\min \; \sum_{j \in J} c_j x_j$$
$$\sum_{j \in J} \boxed{A_j} x_j = b \qquad (\lambda)$$
$$x_j \geq 0$$

Column j of A

Solving an LP by column generation

Suppose the LP relaxation of an integer programming problem has a huge number of variables:

$$\begin{aligned} &\min\ cx \\ &Ax = b \\ &x \geq 0 \end{aligned}$$

We will solve a **restricted master problem**, which has a small subset of the variables:

$$\begin{aligned} &\min\ \sum_{j \in J} c_j x_j \\ &\sum_{j \in J} \boxed{A_j} x_j = b \qquad (\lambda) \\ &x_j \geq 0 \end{aligned}$$

Column j of A

Adding x_k to the problem would improve the solution if x_k has a negative **reduced cost**:

$$r_k = c_k - \boxed{\lambda} A_k < 0$$

Row vector of dual (Lagrange) multipliers

Column generation

Adding x_k to the problem would improve the solution if x_k has a negative reduced cost:

$$r_k = c_k - \lambda A_k < 0$$

Computing the reduced cost of x_k is known as **pricing** x_k.

Cost of column y

So we solve the **pricing problem**:

$$\begin{array}{l} \min\ \boxed{c_y} - \lambda y \\ y \text{ is a column of } A \end{array}$$

Column generation

Adding x_k to the problem would improve the solution if x_k has a negative reduced cost:

$$r_k = c_k - \lambda A_k < 0$$

Computing the reduced cost of x_k is known as **pricing** x_k.

Cost of column y

So we solve the **pricing problem**:

$$\begin{array}{l} \min \ \boxed{c_y} - \lambda y \\ y \text{ is a column of } A \end{array}$$

This can often be solved by **CP**.

We hope to find an optimal solution before generating too many columns.

Airline Crew Scheduling

We want to assign crew members to flights to minimize cost while covering the flights and observing complex work rules.

Flight data

j	s_j	f_j
1	0	3
2	1	3
3	5	8
4	6	9
5	10	12
6	14	16

Start time   Finish time

A **roster** is the sequence of flights assigned to a single crew member.

The gap between two consecutive flights in a roster must be from 2 to 3 hours. Total flight time for a roster must be between 6 and 10 hours.

For example,

flight 1 cannot immediately precede 6

flight 4 cannot immediately precede 5.

The possible rosters are:

(1,3,5), (1,4,6), (2,3,5), (2,4,6)

Airline Crew Scheduling

There are 2 crew members, and the possible rosters are:

1	2	3	4
(1,3,5),	(1,4,6),	(2,3,5),	(2,4,6)

The LP relaxation of the problem is:

Cost of assigning crew member 1 to roster 2

$$\min z$$

$$\begin{bmatrix} 10 & 12 & 7 & 13 & 9 & 11 & 6 & 12 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \end{bmatrix} \begin{matrix} = \\ = \\ = \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \end{matrix} \begin{bmatrix} z \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$x_{ik} \geq 0, \ \text{all } i, k$$

= 1 if we assign crew member 1 to roster 2, = 0 otherwise.

Each crew member is assigned to exactly 1 roster.

Each flight is assigned at least 1 crew member.

Airline Crew Scheduling

There are 2 crew members, and the possible rosters are:

1 2 3 4

(1,3,5), (1,4,6), (2,3,5), (2,4,6)

The LP relaxation of the problem is:

Cost of assigning crew member 1 to roster 2

$$
\min z
$$

$$
\begin{bmatrix}
10 & 12 & 7 & 13 & 9 & 11 & 6 & 12 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1
\end{bmatrix}
\begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \end{bmatrix}
\begin{matrix} = \\ = \\ = \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \end{matrix}
\begin{bmatrix} z \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}
$$

$$
x_{ik} \geq 0, \text{ all } i, k
$$

= 1 if we assign crew member 1 to roster 2, = 0 otherwise.

Each crew member is assigned to exactly 1 roster.

Each flight is assigned at least 1 crew member.

Rosters that cover flight 1.

Airline Crew Scheduling

There are 2 crew members, and the possible rosters are:

1 2 3 4

(1,3,5), (1,4,6), (2,3,5), (2,4,6)

The LP relaxation of the problem is:

Cost of assigning crew member 1 to roster 2

min z

$$\begin{bmatrix} 10 & 12 & 7 & 13 & 9 & 11 & 6 & 12 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \end{bmatrix} \begin{matrix} = \\ = \\ = \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \end{matrix} \begin{bmatrix} z \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$x_{ik} \geq 0$, all i, k

= 1 if we assign crew member 1 to roster 2, = 0 otherwise.

Each crew member is assigned to exactly 1 roster.

Each flight is assigned at least 1 crew member.

Rosters that cover flight 2.

Airline Crew Scheduling

There are 2 crew members, and the possible rosters are:

1	2	3	4
(1,3,5),	(1,4,6),	(2,3,5),	(2,4,6)

The LP relaxation of the problem is:

Cost of assigning crew member 1 to roster 2

$$\min z$$

$$\begin{bmatrix} 10 & 12 & 7 & 13 & 9 & 11 & 6 & 12 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ \boxed{1} & 0 & \boxed{1} & 0 & \boxed{1} & 0 & \boxed{1} & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \end{bmatrix} \begin{matrix} = \\ = \\ = \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \end{matrix} \begin{bmatrix} z \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$x_{ik} \geq 0, \ \text{all } i, k$$

= 1 if we assign crew member 1 to roster 2, = 0 otherwise.

Each crew member is assigned to exactly 1 roster.

Each flight is assigned at least 1 crew member.

Rosters that cover flight 3.

Airline Crew Scheduling

There are 2 crew members, and the possible rosters are:

$$\begin{array}{cccc} 1 & 2 & 3 & 4 \\ (1,3,5), & (1,4,6), & (2,3,5), & (2,4,6) \end{array}$$

The LP relaxation of the problem is:

Cost of assigning crew member 1 to roster 2

$$\min z$$

$$\begin{bmatrix} 10 & 12 & 7 & 13 & 9 & 11 & 6 & 12 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & \boxed{1} & 0 & \boxed{1} & 0 & \boxed{1} & 0 & \boxed{1} \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \end{bmatrix} \begin{matrix} = \\ = \\ = \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \end{matrix} \begin{bmatrix} z \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$x_{ik} \geq 0, \text{ all } i, k$$

= 1 if we assign crew member 1 to roster 2, = 0 otherwise.

Each crew member is assigned to exactly 1 roster.

Each flight is assigned at least 1 crew member.

Rosters that cover flight 4.

Airline Crew Scheduling

There are 2 crew members, and the possible rosters are:

1 2 3 4

(1,3,5), (1,4,6), (2,3,5), (2,4,6)

The LP relaxation of the problem is:

Cost of assigning crew member 1 to roster 2

= 1 if we assign crew member 1 to roster 2, = 0 otherwise.

$$
\min z
$$

$$
\begin{bmatrix}
10 & 12 & 7 & 13 & 9 & 11 & 6 & 12 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
\boxed{1} & 0 & \boxed{1} & 0 & \boxed{1} & 0 & \boxed{1} & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{24}
\end{bmatrix}
\begin{matrix}
= \\ = \\ = \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq
\end{matrix}
\begin{bmatrix}
z \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1
\end{bmatrix}
$$

$$
x_{ik} \geq 0, \ \text{all } i, k
$$

Each crew member is assigned to exactly 1 roster.

Each flight is assigned at least 1 crew member.

Rosters that cover flight 5.

Airline Crew Scheduling

There are 2 crew members, and the possible rosters are:

1 2 3 4

(1,3,5), (1,4,6), (2,3,5), (2,4,6)

The LP relaxation of the problem is:

Cost of assigning crew member 1 to roster 2

$$
\min z
$$

$$
\begin{bmatrix}
10 & 12 & 7 & 13 & 9 & 11 & 6 & 12 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & \boxed{1} & 0 & \boxed{1} & 0 & \boxed{1} & 0 & \boxed{1}
\end{bmatrix}
\begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \end{bmatrix}
\begin{matrix} = \\ = \\ = \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \end{matrix}
\begin{bmatrix} z \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}
$$

$$
x_{ik} \geq 0, \text{ all } i, k
$$

= 1 if we assign crew member 1 to roster 2, = 0 otherwise.

Each crew member is assigned to exactly 1 roster.

Each flight is assigned at least 1 crew member.

Rosters that cover flight 6.

Airline Crew Scheduling

There are 2 crew members, and the possible rosters are:

1	2	3	4
(1,3,5),	(1,4,6),	(2,3,5),	(2,4,6)

The LP relaxation of the problem is:

Cost c_{12} of assigning crew member 1 to roster 2

$$\min z$$

$$\begin{bmatrix} 10 & 12 & 7 & 13 & 9 & 11 & 6 & 12 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & \boxed{1} & 0 & \boxed{1} & 0 & \boxed{1} & 0 & \boxed{1} \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \end{bmatrix} \begin{matrix} = \\ = \\ = \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \end{matrix} \begin{bmatrix} z \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$x_{ik} \geq 0, \text{ all } i, k$$

= 1 if we assign crew member 1 to roster 2, = 0 otherwise.

Each crew member is assigned to exactly 1 roster.

Each flight is assigned at least 1 crew member.

In a real problem, there can be **millions** of rosters.

Airline Crew Scheduling

We start by solving the problem with a subset of the columns:

Optimal dual solution

$$\min\ z$$

$$\begin{bmatrix} 10 & 13 & 9 & 12 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{14} \\ x_{21} \\ x_{24} \end{bmatrix} \begin{matrix} = \\ = \\ = \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \end{matrix} \begin{bmatrix} z \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \qquad \boxed{\begin{matrix} (10) \\ (9) \\ (0) \\ (0) \\ (0) \\ (0) \\ (0) \\ (3) \end{matrix}} \begin{matrix} u_1 \\ u_2 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{matrix}$$

$$x_{ik} \geq 0,\ \text{all } i, k$$

Airline Crew Scheduling

We start by solving the problem with a subset of the columns:

Dual variables

$$\min\ z$$

$$\begin{bmatrix} 10 & 13 & 9 & 12 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{14} \\ x_{21} \\ x_{24} \end{bmatrix} \begin{array}{c} = \\ = \\ = \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \end{array} \begin{bmatrix} z \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \qquad \begin{array}{c} \\ (10) \\ (9) \\ (0) \\ (0) \\ (0) \\ (0) \\ (0) \\ (3) \end{array} \ \boxed{\begin{array}{c} u_1 \\ u_2 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{array}}$$

$$x_{ik} \geq 0, \ \text{all } i, k$$

Airline Crew Scheduling

We start by solving the problem with a subset of the columns:

$$\min\ z$$

$$\begin{bmatrix} 10 & 13 & 9 & 12 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{14} \\ x_{21} \\ x_{24} \end{bmatrix} \begin{matrix} = \\ = \\ = \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \end{matrix} \begin{bmatrix} z \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \qquad \begin{matrix} \\ (10) & u_1 \\ (9) & u_2 \\ (0) & v_1 \\ (0) & v_2 \\ (0) & v_3 \\ (0) & v_4 \\ (0) & v_5 \\ (3) & v_6 \end{matrix}$$

$$x_{ik} \geq 0, \text{ all } i, k$$

Dual variables

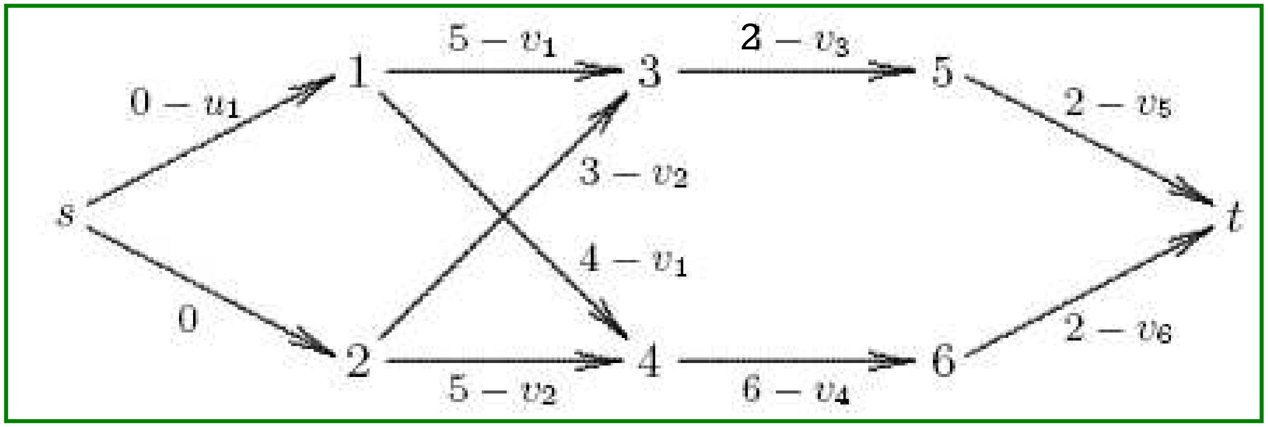
The reduced cost of an excluded roster *k* for crew member *i* is

$$c_{ik} - u_i - \sum_{j \text{ in roster k}} v_j$$

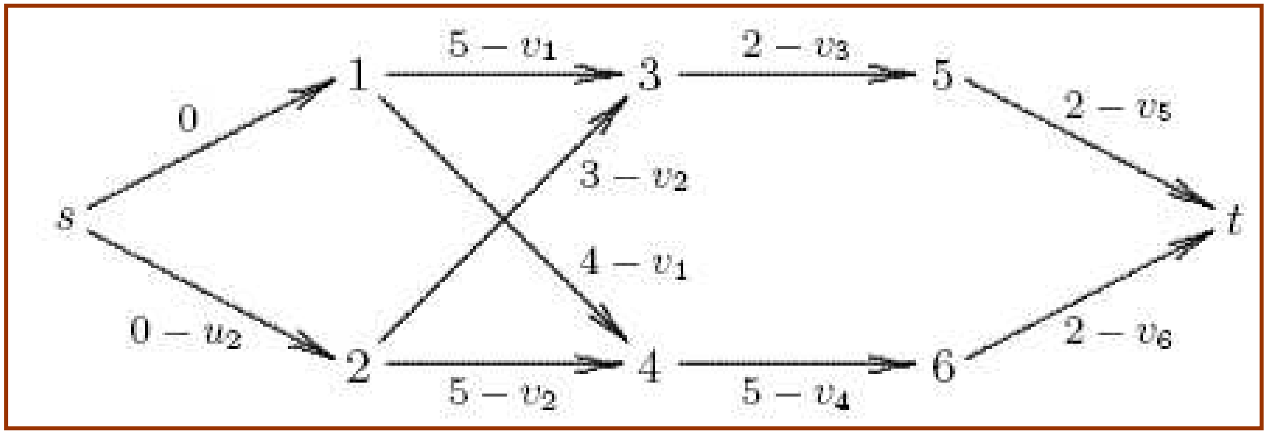
We will formulate the pricing problem as a shortest path problem.

Pricing problem

Crew member 1



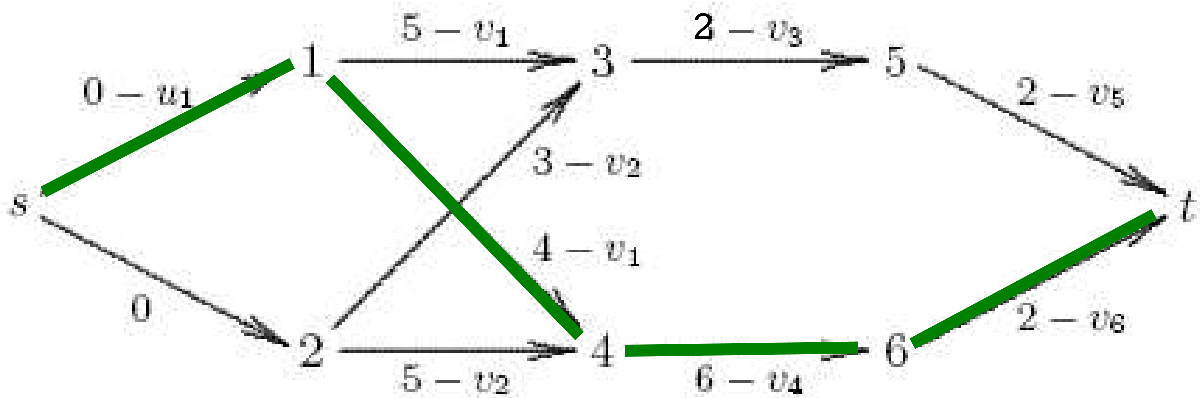
Crew member 2



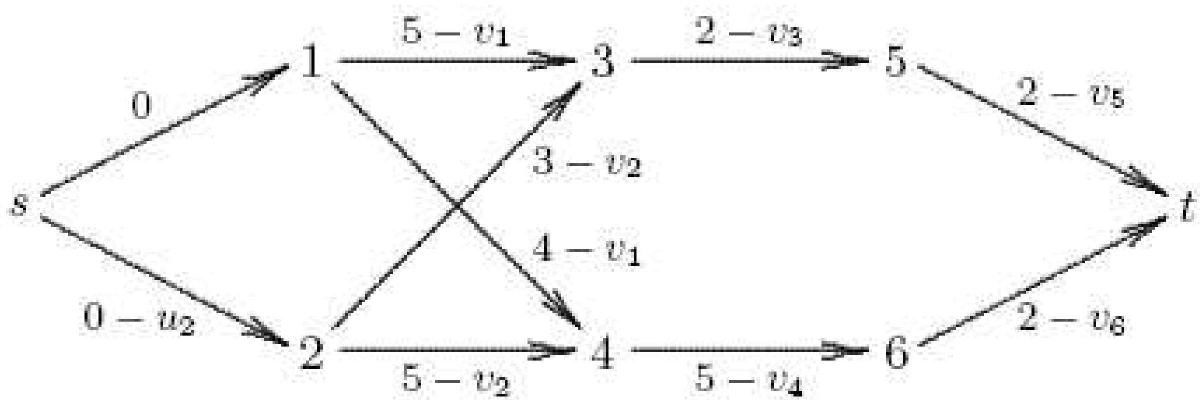
Pricing problem

Each s-t path corresponds to a roster, provided the flight time is within bounds.

Crew member 1



Crew member 2

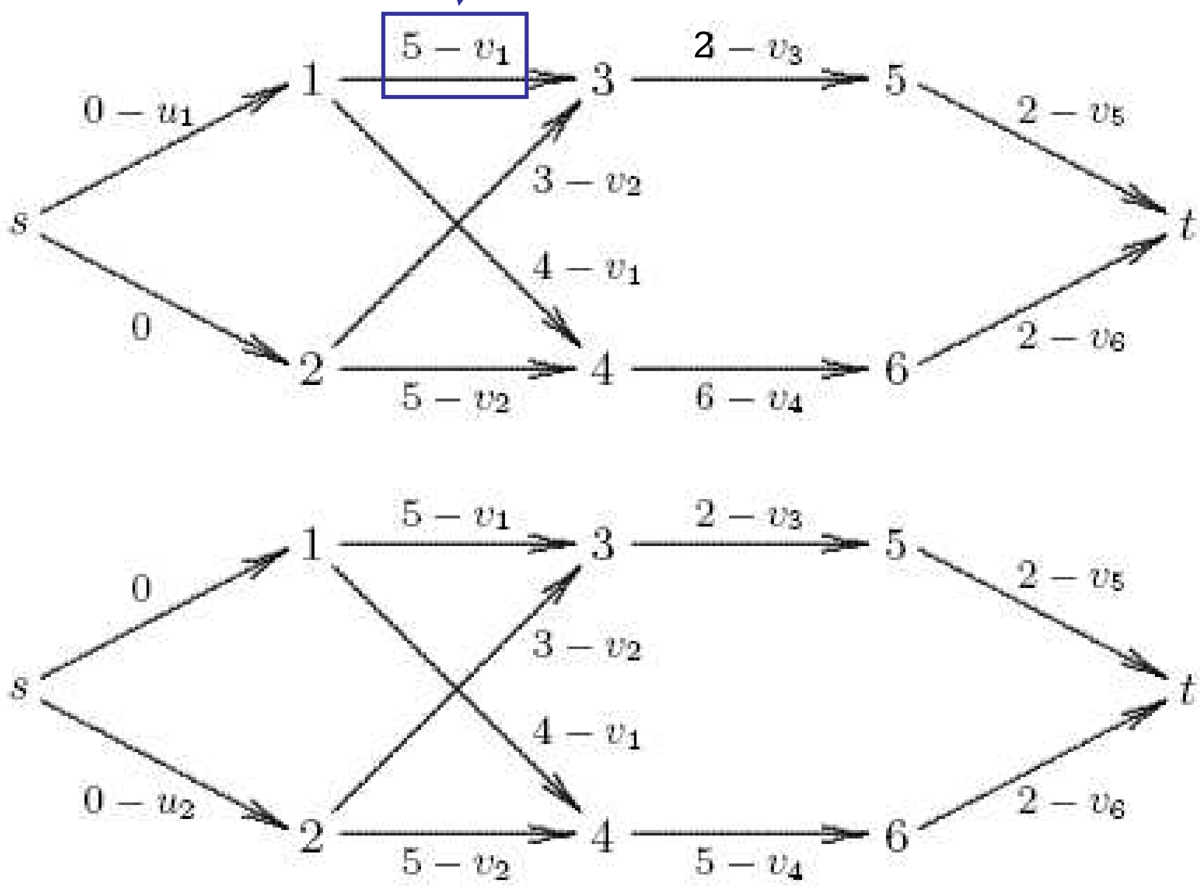


Pricing problem

Cost of flight 3 if it immediately follows flight 1, offset by dual multiplier for flight 1

Crew member 1

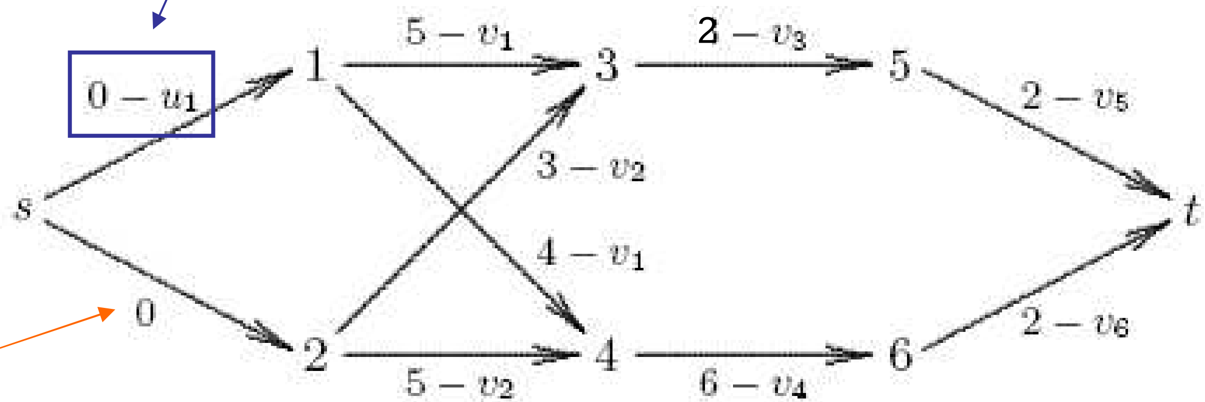
Crew member 2



Pricing problem

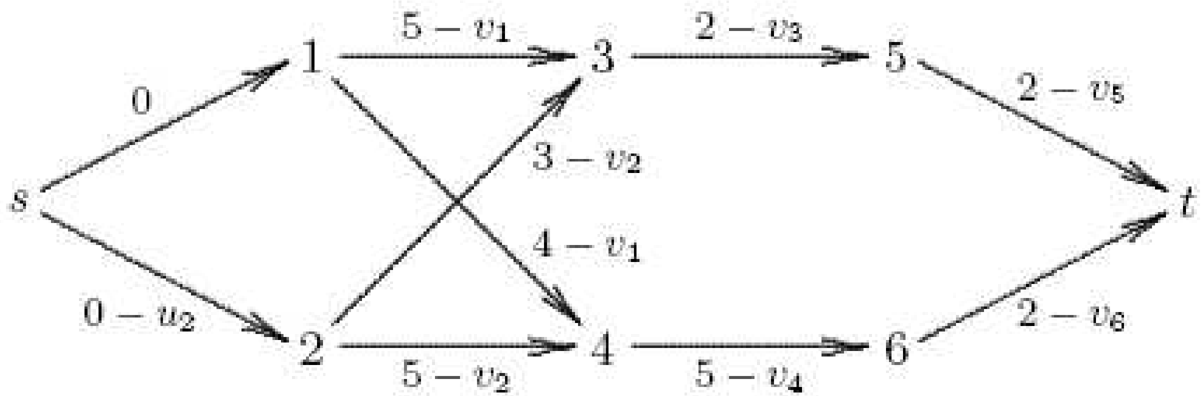
Cost of transferring from home to flight 1, offset by dual multiplier for crew member 1

Crew member 1



Dual multiplier omitted to break symmetry

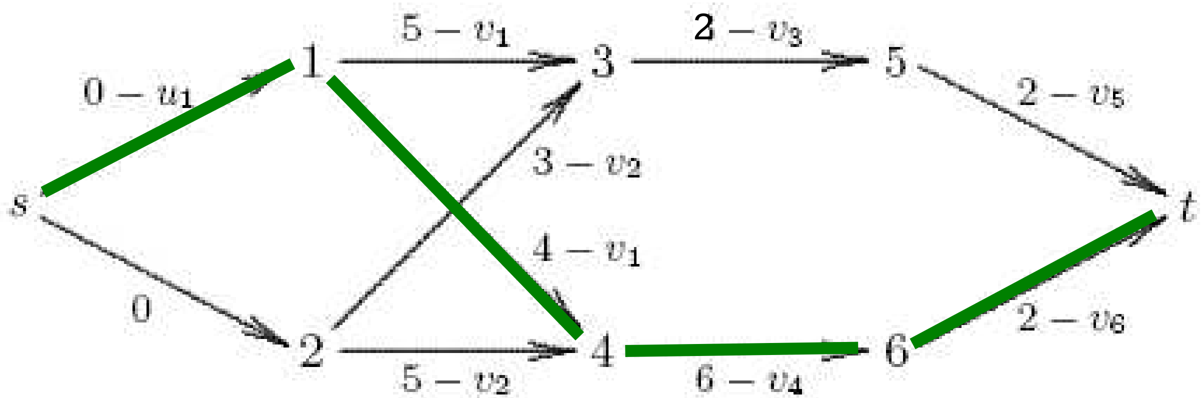
Crew member 2



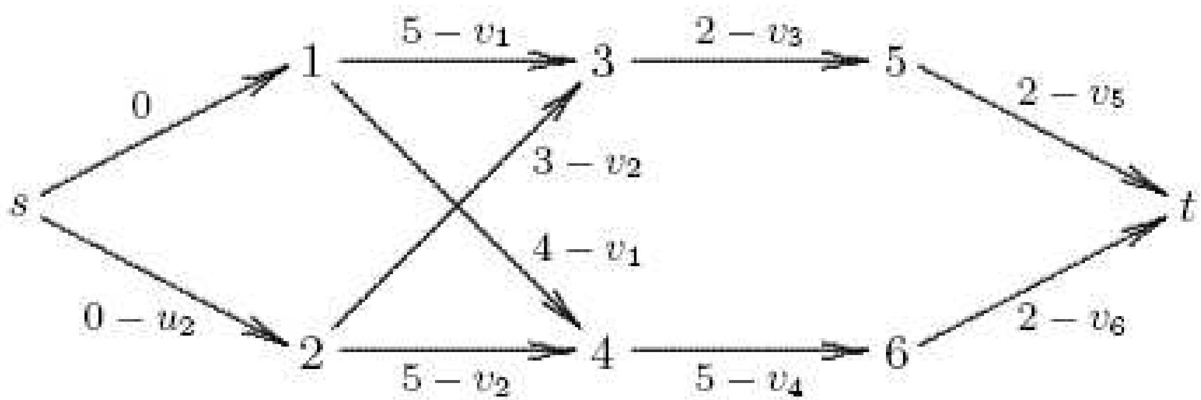
Pricing problem

Length of a path is reduced cost of the corresponding roster.

Crew member 1



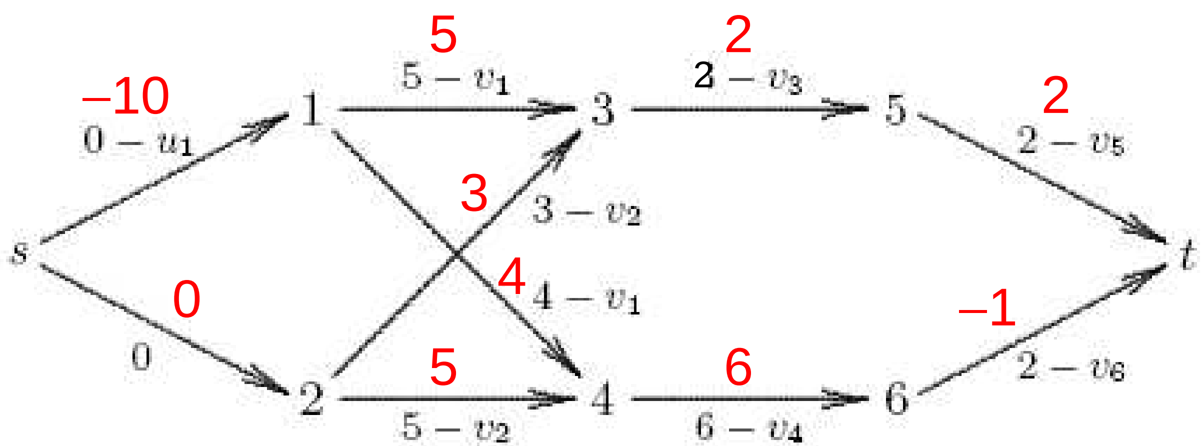
Crew member 2



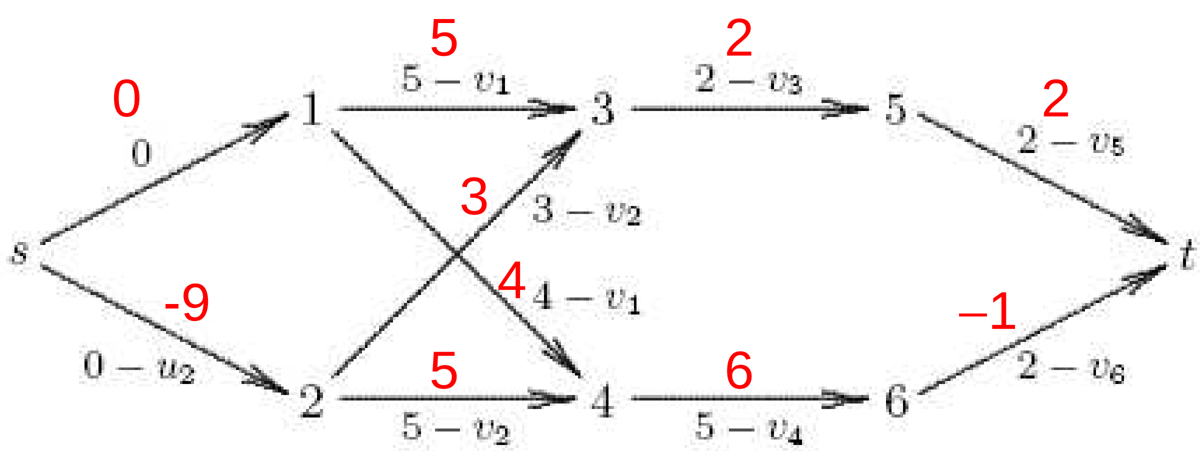
Pricing problem

Arc lengths using dual solution of LP relaxation

Crew member 1



Crew member 2

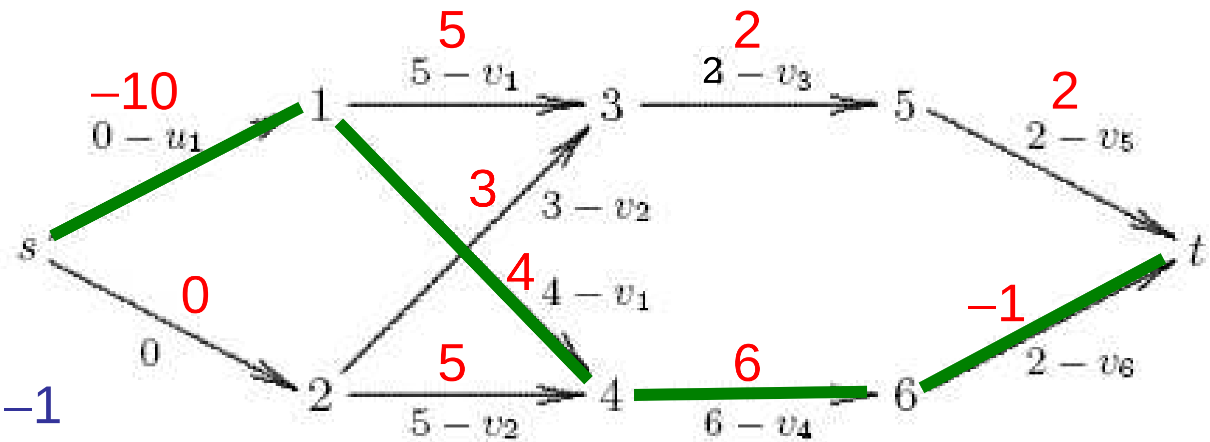


Pricing problem

Solution of shortest path problems

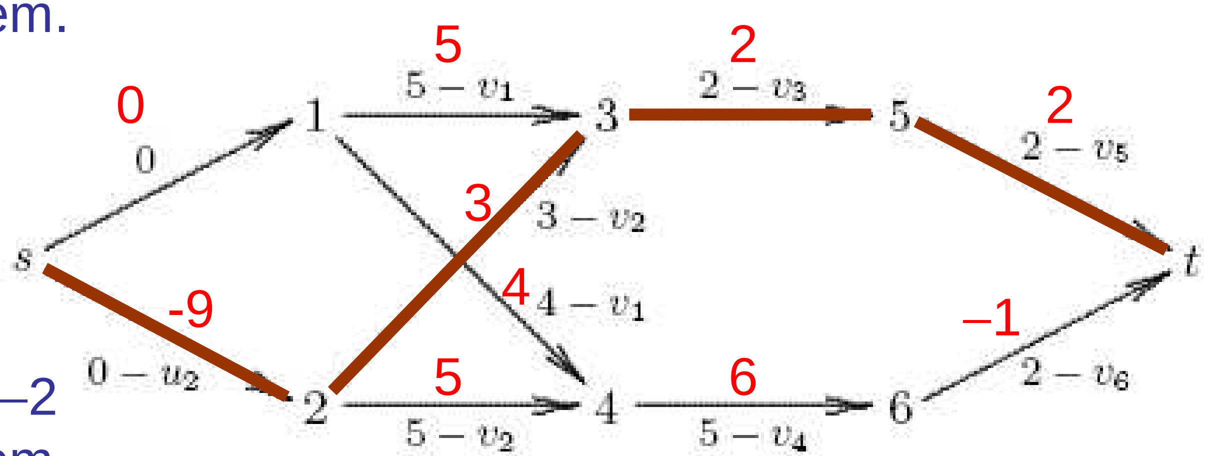
Crew member 1

Reduced cost = –1
Add x_{12} to problem.



Crew member 2

Reduced cost = –2
Add x_{23} to problem.



After x_{12} and x_{23} are added to the problem, no remaining variable has negative reduced cost.

Pricing problem

The shortest path problem cannot be solved by traditional shortest path algorithms, due to the bounds on total path length.

It **can** be solved by CP:

Set of flights assigned to crew member *i*

Path length

Graph

Path global constraint

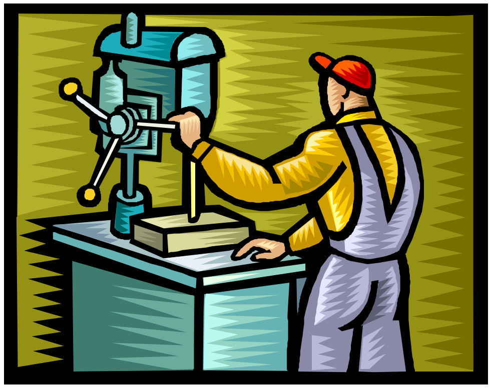
$$\text{Path}(\boxed{X_i}, \boxed{z_i}, \boxed{G}), \text{ all flights } i$$

Setsum global constraint

$$T_{\min} \le \sum_{j \in X_i} \left(\boxed{f_j - s_j} \right) \le T_{\max}$$

$$X_i \subset \{\text{flights}\}, \quad z_i < 0, \text{ all } i$$

Duration of flight *j*

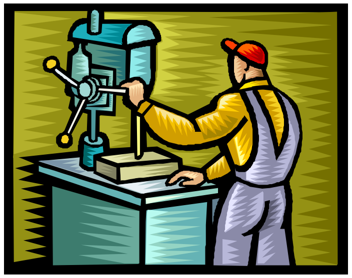


Example: Machine Scheduling

Constraint-directed search using
logic-based Benders decomposition

This example illustrates:

- **Constraint-based search** as Benders decomposition
  - Nogoods are **Benders cuts**.
- Solution of the **master problem** by **MILP**.
  - Allocate jobs to machines.
- Solution of the **subproblem** by **CP**.
  - Schedule jobs on each machine



Benders decomposition

Constraint-directed search in which the master problem contains a fixed set of variables *x*.

Applied to problems of the form

$$\begin{aligned} &\min \ f(x,y) \\ &S(x,y) \\ &x \in D_x,\ \ y \in D_y \end{aligned}$$

When x is fixed to some value, the resulting **subproblem** is much easier:

$$\begin{aligned} &\min \ f(\bar{x},y) \\ &S(\bar{x},y) \\ &y \in D_y \end{aligned}$$

…perhaps because it decouples into smaller problems.

Benders decomposition

Constraint-directed search in which the master problem contains a fixed set of variables *x*.

Applied to problems of the form

$$\min \ f(x,y)$$
$$S(x,y)$$
$$x \in D_x,\ \ y \in D_y$$

When x is fixed to some value, the resulting **subproblem** is much easier:

$$\min \ f(\bar{x},y)$$
$$S(\bar{x},y)$$
$$y \in D_y$$

…perhaps because it decouples into smaller problems.

Nogoods are **Benders cuts** and exclude solutions no better than *x.*

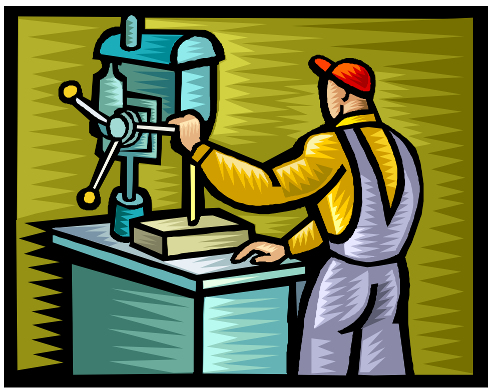
The Benders cut is obtained by solving the **inference dual** of the subproblem (**classically**, the linear programming dual).

Machine Scheduling

• Assign 5 jobs to 2 machines (A and B), and schedule the machines assigned to each machine within time windows.

• The objective is to minimize **makespan**.

Time lapse between start of first job and end of last job.



• Assign the jobs in the **master problem**, to be solved by **MILP**.

• Schedule the jobs in the **subproblem**, to be solved by **CP**.

Machine Scheduling

Job Data

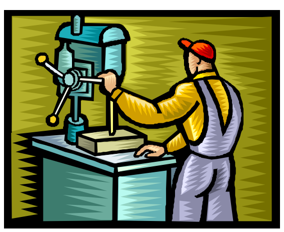
Job j	Release time r_j	Dead-line d_j	Processing time p_{Aj}	p_{Bj}
1	0	10	1	5
2	0	10	3	6
3	2	7	3	7
4	2	10	4	6
5	4	7	2	5

Machine A

Machine B

Once jobs are assigned, we can minimize overall makespan by minimizing makespan on each machine individually.

So the subproblem decouples.



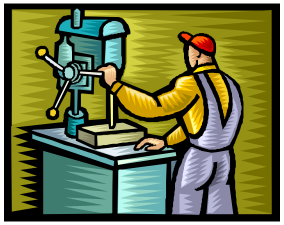
Machine Scheduling

Job Data

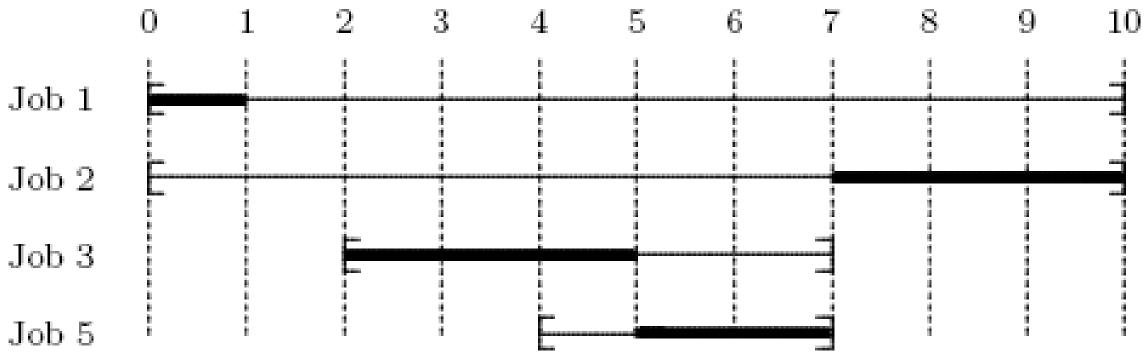
Job j	Release time r_j	Dead-line d_j	Processing time p_{Aj}	p_{Bj}
1	0	10	1	5
2	0	10	3	6
3	2	7	3	7
4	2	10	4	6
5	4	7	2	5

Once jobs are assigned, we can minimize overall makespan by minimizing makespan on each machine individually.

So the subproblem decouples.



Minimum makespan schedule for jobs 1, 2, 3, 5 on machine A



Machine Scheduling

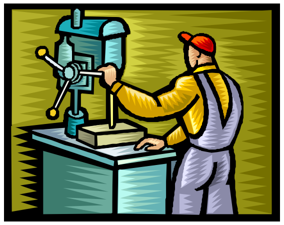
The problem is

Start time of job *j*

Time windows

Jobs cannot overlap

$$
\begin{aligned}
&\min \; M \\
&M \ge \boxed{s_j} + p_{x_j j}, \text{ all } j \\
&r_j \le s_j \le d_j - p_{x_j j}, \text{ all } j \\
&\text{disjunctive}\big((s_j \mid x_j = i), (p_{ij} \mid x_j = i)\big), \text{ all } i
\end{aligned}
$$

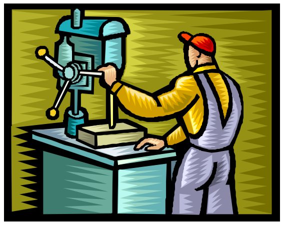


Machine Scheduling

The problem is

Indexed linear metaconstraint

$$\begin{aligned}
&\min\ M \\
&\boxed{\begin{aligned} &M \geq s_j + p_{x_j j},\ \text{all } j \\ &r_j \leq s_j \leq d_j - p_{x_j j},\ \text{all } j \end{aligned}} \\
&\text{disjunctive}\big((s_j \,|\, x_j = i), (p_{ij} \,|\, x_j = i)\big),\ \text{all } i
\end{aligned}$$



Machine Scheduling

The problem is

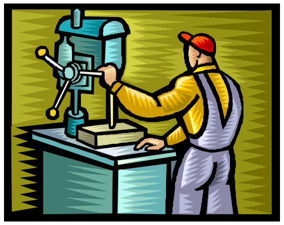
$$\min \; M$$

$$\boxed{\begin{array}{l} M \ge s_j + p_{x_j j}, \text{ all } j \\ r_j \le s_j \le d_j - p_{x_j j}, \text{ all } j \end{array}}$$

Specially-structured indexed linear metaconstraint

$$\boxed{\text{disjunctive}\big((s_j \,|\, x_j = i), (p_{ij} \,|\, x_j = i)\big), \text{ all } i}$$

Disjunctive scheduling metaconstraint



Machine Scheduling

The problem is

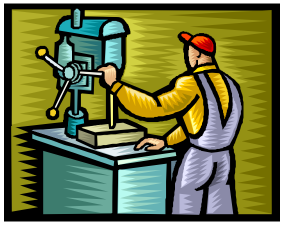
Start time of job j

$$\min \ M$$
$$M \geq \boxed{s_j} + p_{x_j j}, \ \text{all } j$$
$$r_j \leq s_j \leq d_j - p_{x_j j}, \ \text{all } j$$
$$\text{disjunctive}\Big((s_j \,|\, x_j = i), (p_{ij} \,|\, x_j = i) \Big), \ \text{all } i$$

Time windows

Jobs cannot overlap

For a fixed assignment $\bar{x}$ the subproblem on each machine i is

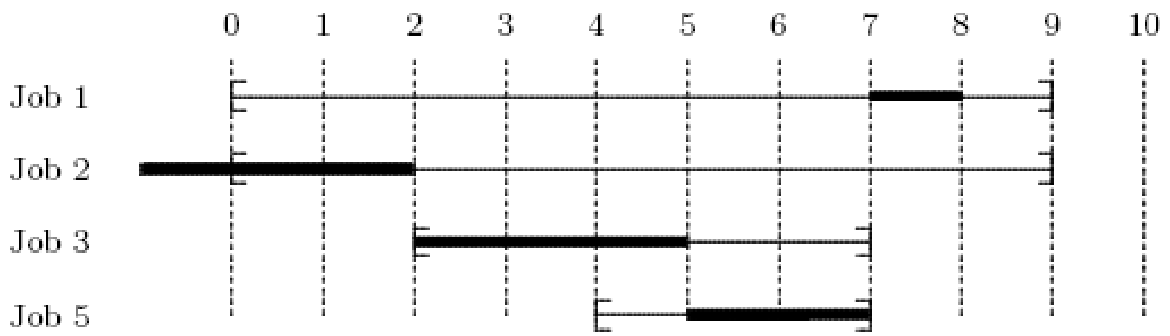


$$\min \ M$$
$$M \geq s_j + p_{\bar{x}_j j}, \ \text{all } j \text{ with } \bar{x}_j = i$$
$$r_j \leq s_j \leq d_j - p_{\bar{x}_j j}, \ \text{all } j \text{ with } \bar{x}_j = i$$
$$\text{disjunctive}\Big((s_j \,|\, \bar{x}_j = i), (p_{ij} \,|\, \bar{x}_j = i) \Big)$$

Benders cuts

Suppose we assign jobs 1,2,3,5 to machine A in iteration *k*.

We can prove that 10 is the optimal makespan by proving that the schedule is infeasible with makespan 9.



Edge finding derives infeasibility by reasoning only with jobs 2,3,5. So these jobs alone create a minimum makespan of 10.

So we have a Benders cut

$$v \geq B_{k+1}(x) = \begin{cases} 10 & \text{if } x_2 = x_3 = x_4 = A \\ 0 & \text{otherwise} \end{cases}$$

Benders cuts

We want the master problem to be an MILP, which is good for assignment problems.

So we write the Benders cut

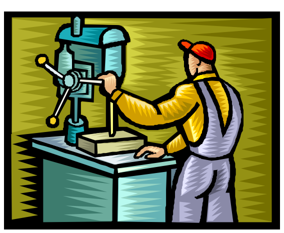
$$v \geq B_{k+1}(x) = \begin{cases} 10 & \text{if } x_2 = x_3 = x_4 = A \\ 0 & \text{otherwise} \end{cases}$$

Using 0-1 variables:

$$v \geq 10\left(x_{A2} + x_{A3} + \boxed{x_{A5}} - 2\right)$$

$$v \geq 0$$

= 1 if job 5 is assigned to machine A



Master problem

The master problem is an MILP:

$$\min \; v$$

$$\sum_{j=1}^{5} p_{Aj} x_{Aj} \le 10, \text{ etc.}$$

$$\sum_{j=1}^{5} p_{Bj} x_{Bj} \le 10, \text{ etc.}$$

$$v \ge \sum_{j=1}^{5} p_{ij} x_{ij}, \quad v \ge 2 + \sum_{j=3}^{5} p_{ij} x_{ij}, \text{ etc.}, \quad i = A,B$$

$$\text{v} \ge 10(x_{A2} + x_{A3} + x_{A5} - 2)$$

$$v \ge 8 x_{B4}$$

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

Constraints derived from time windows

Constraints derived from release times

Benders cut from machine A

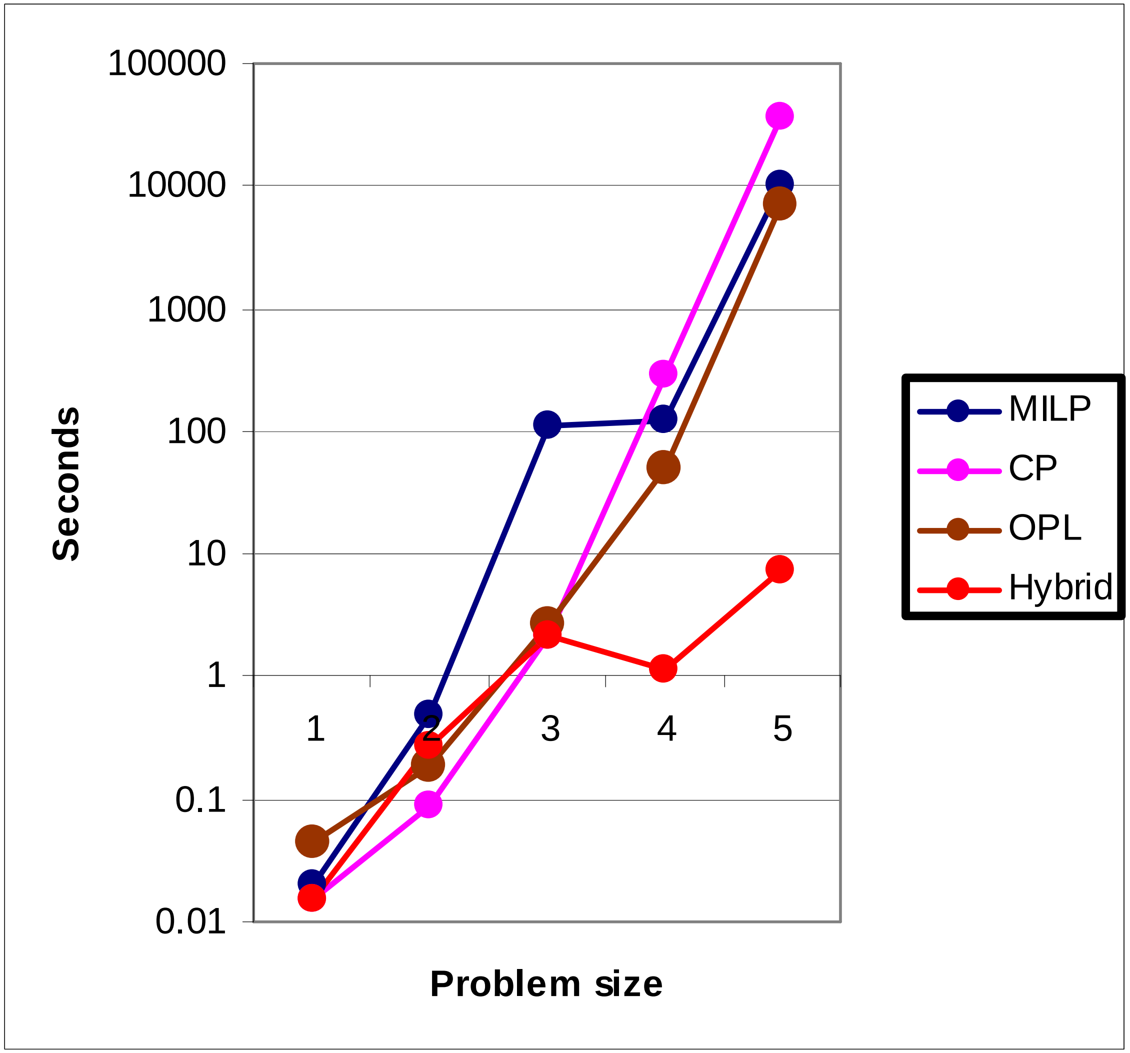
Benders cut from machine B

Computational Results

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



Stronger Benders cuts

If all release times are the same, we can strengthen the Benders cuts.

We are now using the cut

$$v \geq M_{ik} \left(\sum_{j \in J_{ik}} x_{ij} - |J_{ik}| + 1 \right)$$

Min makespan on machine *i* in iteration *k*

Set of jobs assigned to machine *i* in iteration *k*

A stronger cut provides a useful bound even if only some of the jobs in J_{ik} are assigned to machine *i*:

$$v \geq M_{ik} - \sum_{j \in J_{ik}} (1 - x_{ij}) p_{ij}$$

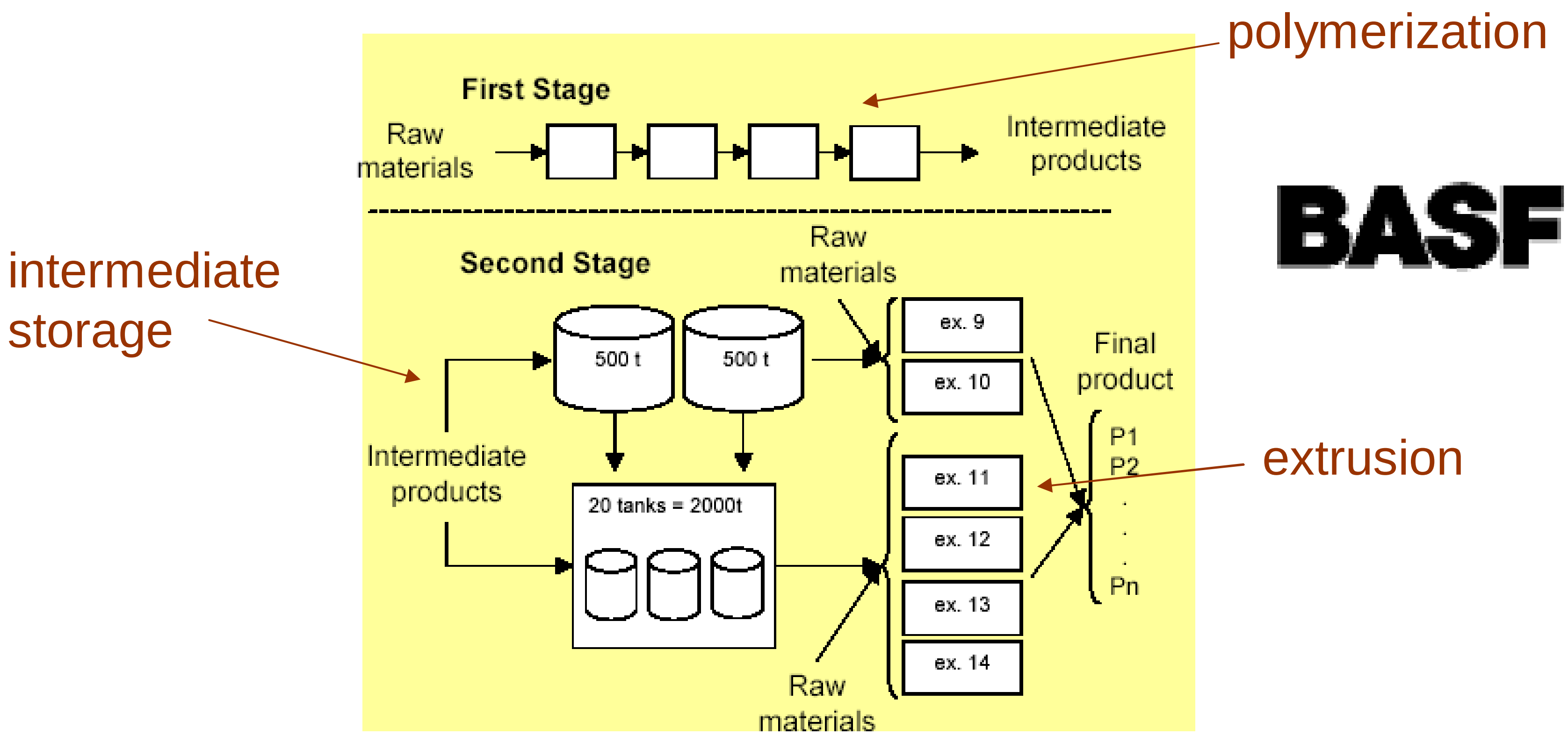
These results can be generalized to **cumulative scheduling**.

Three success stories

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

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

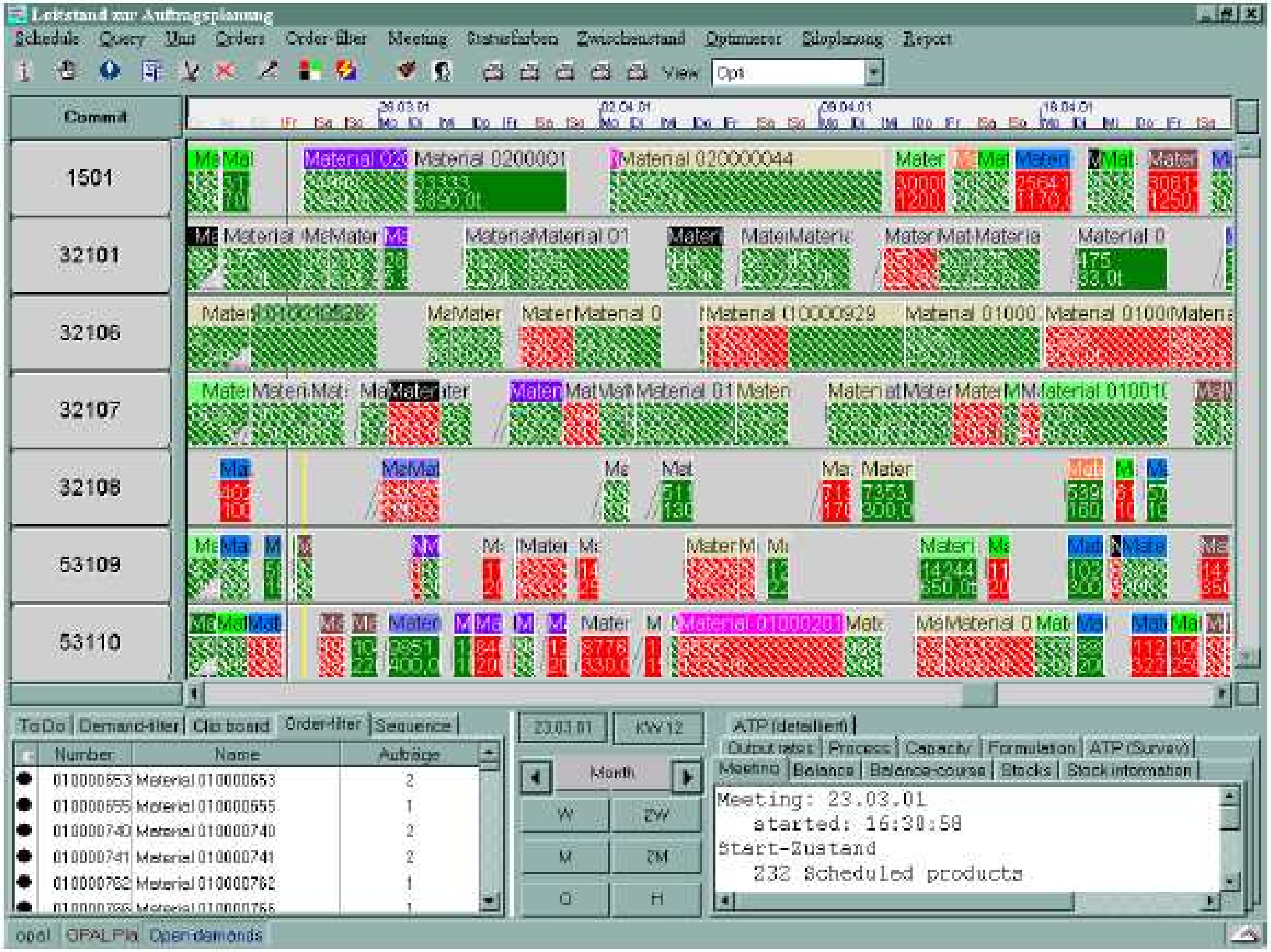
Process scheduling and lot sizing at BASF

- The problems are interdependent
  - *Lot sizing* depends on assignment, because 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.

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

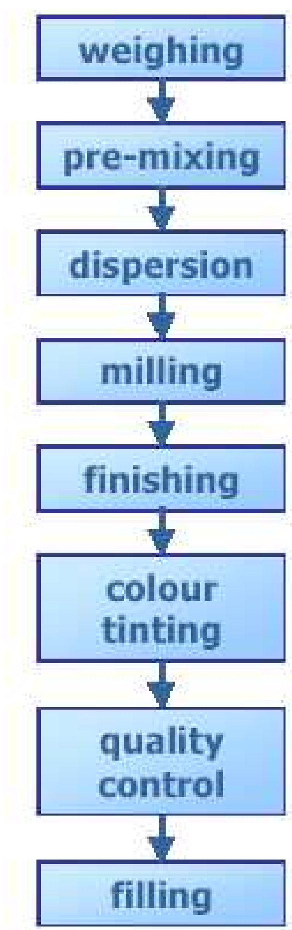


Source: BASF

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



- 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.

Several machines of each type

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.

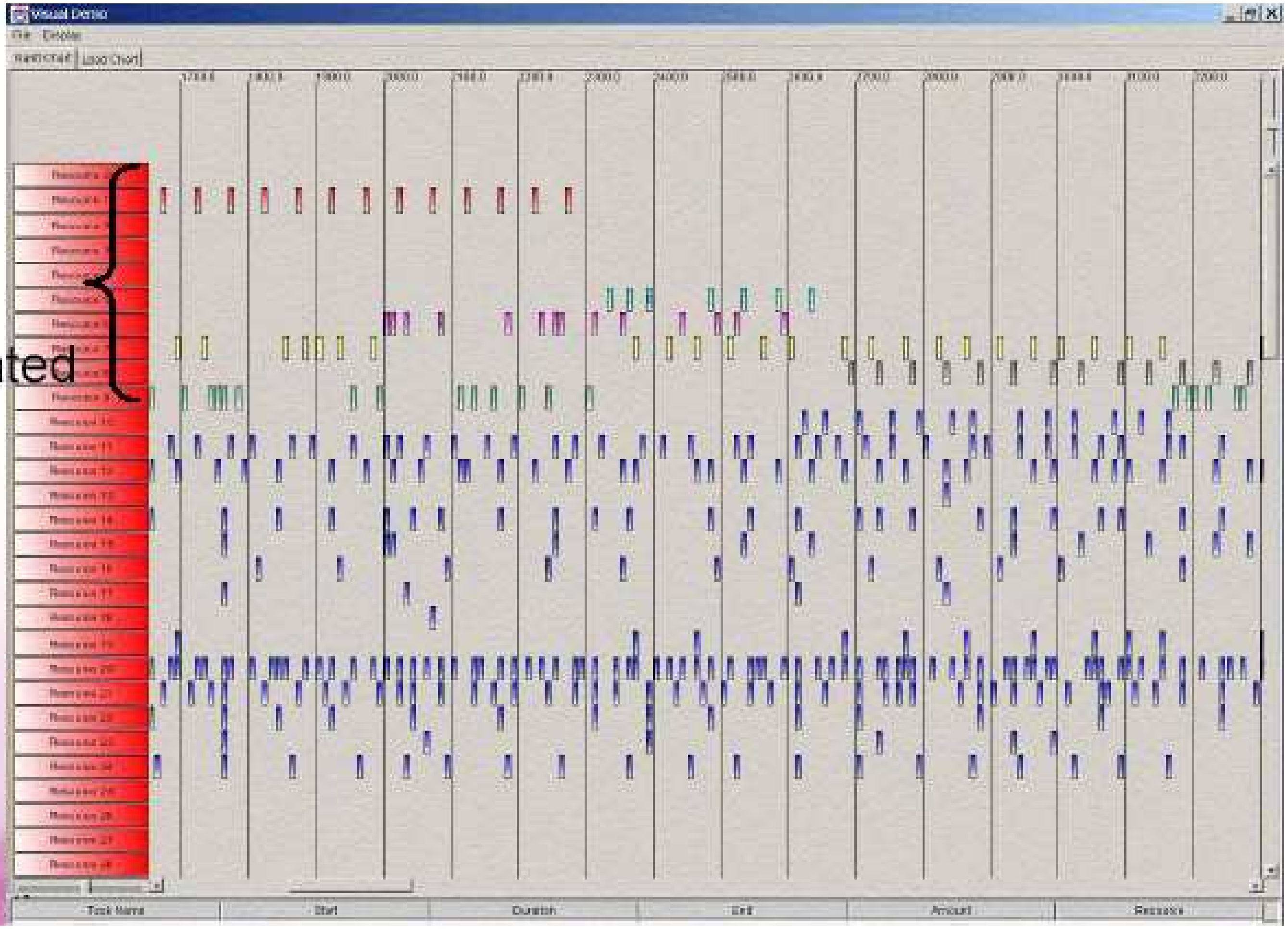
Production line sequencing at Peugeot-Citroën

- 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

Colour

concentrated



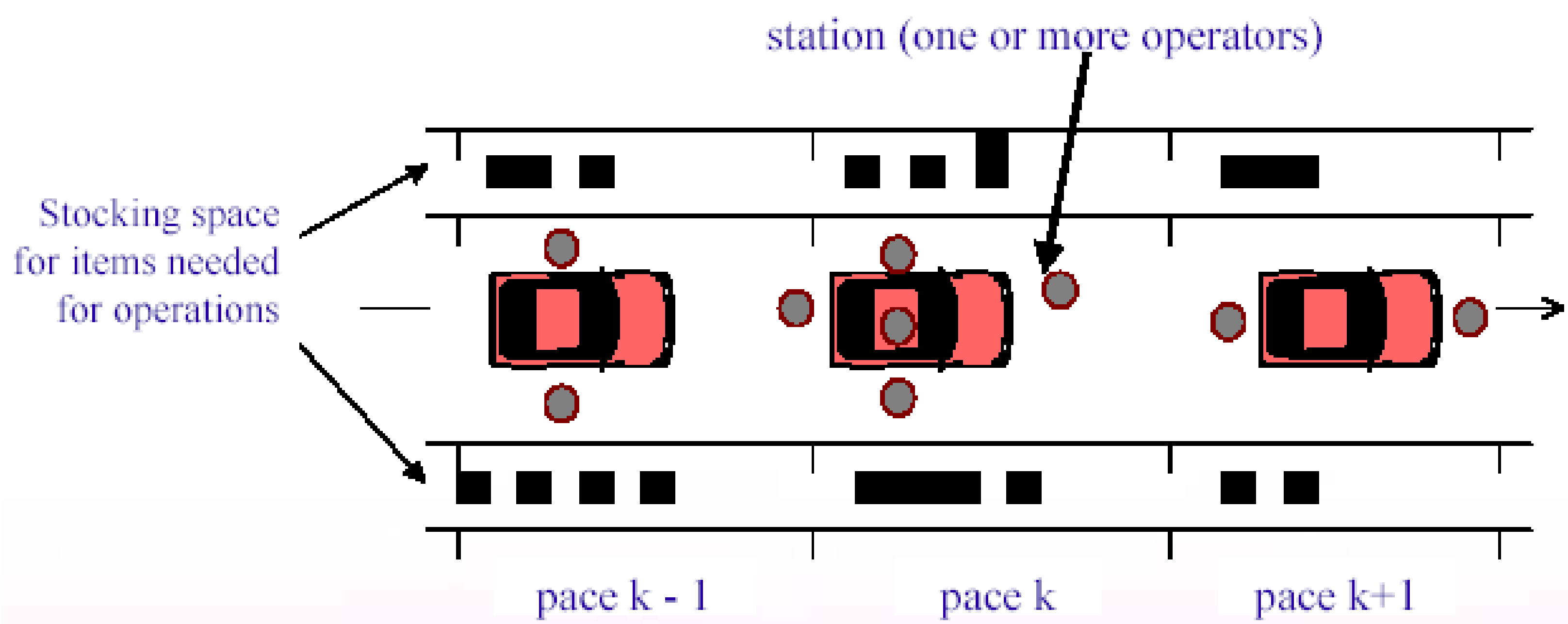
Source: Peugeot/Citroën

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

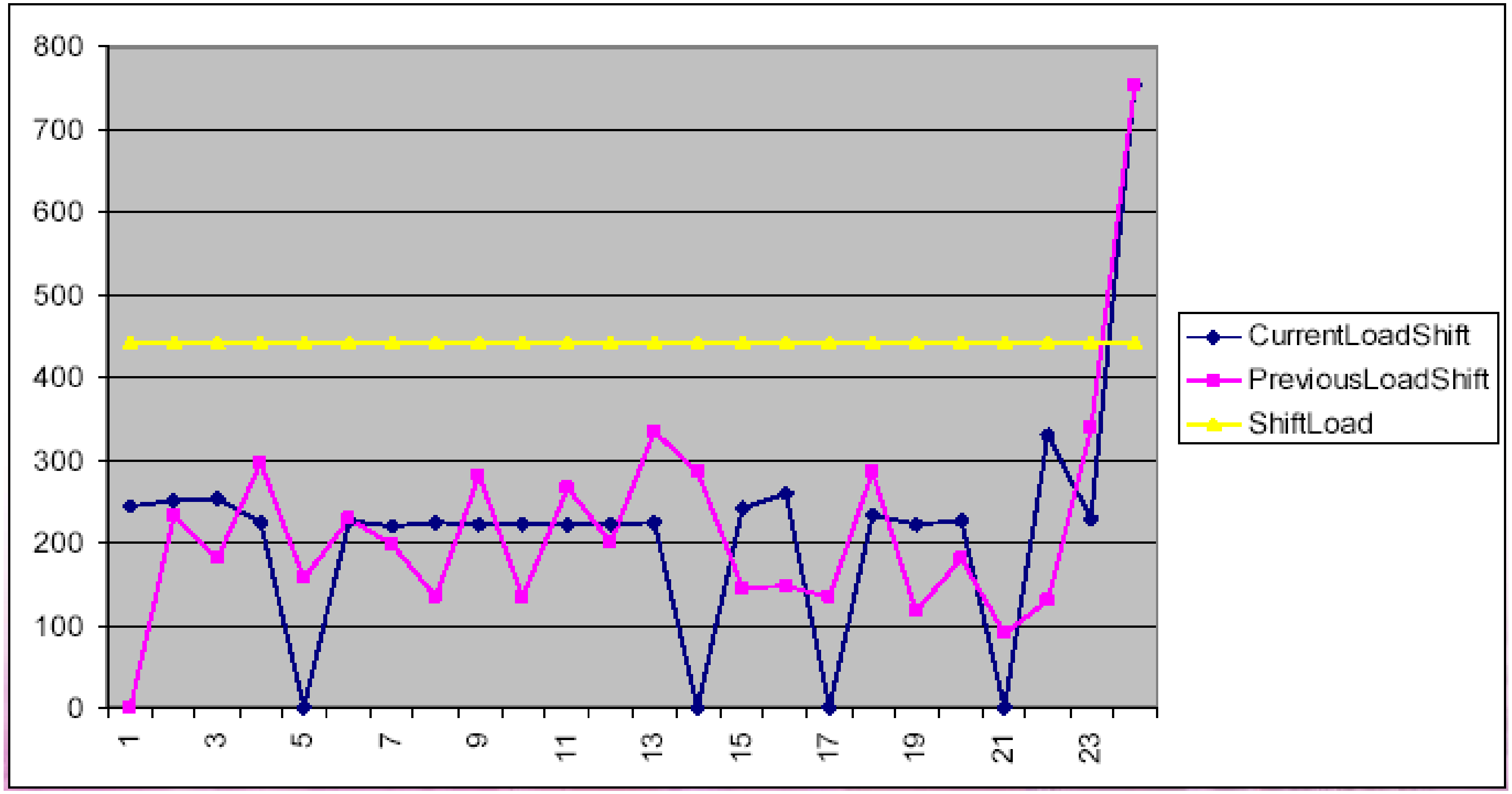
- Objective
  - Equalize load at work stations.
  - Keep each worker on one side of the car

- 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 an integrated 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.

Mathematical Programming Solvers

- Mixed integer programming
  - Commercial
    - CPLEX, OSL, Xpress-MP, Excel solver, LINDO
  - Non-commercial
    - ABACUS, BCP, BonsaiG, CBC, GLPK, lp_solve, MINTO, SCIP, SYMPHONY
- Global optimization
  - BARON, LGO

Mathematical Programming Solvers

- Modeling systems
  - Commercial
    - AMPL, GAMS, AIMMS, OPL Studio
  - Non-commercial
    - ZIMPL, GMPL

CP Systems

- Solvers
  - Commercial
    - ILOG Solver/Scheduler, CHIP, Xpress-Kalis
  - Non-commercial
    - ECLiPSe, SCIP
- Modeling Systems
  - Mosel (runs Xpress-Kalis), ECLiPSe, Mozart, OPL Studio (runs ILOG solver/scheduler)

Integrated Systems

- Cooperating solvers
  - OPL Studio (runs CPLEX, ILOG Solver/Scheduler)
  - ECLiPSe (runs ECLiPSe CP solver, Xpress-MP)
- Integration with low-level modeling
  - Mosel (runs Xpress-MP, Xpress-Kalis)
- Integration with high-level modeling
  - BARON (partial integration for global optimization)
  - SIMPL (just released, non-commercial)


THE
END