Recent Developments in Integrated Methods for Optimization

John Hooker
Carnegie Mellon University

EWO Seminar
August 2012

Premise

- Constraint programming and mathematical programming can work together.
  - There is underlying unity.
  - Result: faster solution and more elegant models.

Premise

- Constraint programming and mathematical programming can work together.
  - There is underlying unity.
  - Result: faster solution and more elegant models.
- Math programming solvers constantly improve…
  - But they would reach a much higher level if this same effort were applied to integrated methods.
  - Solvers are beginning to move in this direction.

Premise

- Constraint programming and mathematical programming can work together.
  - There is underlying unity.
  - Result: faster solution and more elegant models.
- Math programming solvers constantly improve…
  - But they would reach a much higher level if this same effort were applied to integrated methods.
  - Solvers are beginning to move in this direction.
- The same integration principles apply more generally.
  - Global optimization, exact/heuristic methods.

Outline

- What is constraint programming?
- Integrating OR and CP
  - Constraint propagation + relaxation
  - CP-based branch and price
  - Decomposition Methods
  - Software
- Some very recent work

Caveat: This is a high-level overview. Don't worry about the technical details.

What Is Constraint Programming?

Basic Idea

Applications

Examples

CP and Math Programming

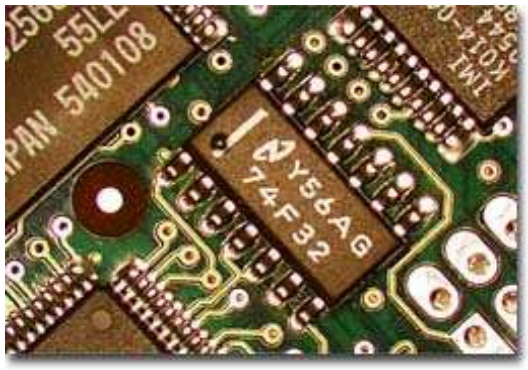
Software

Basic Idea

- Each line of the model is both a **constraint** and a **procedure**.
  - **Constraint:** often a high-level **global constraint**
    - Different modeling paradigm than math programming
  - **Procedure**: removes infeasible values from variable domains
    - Filtering, domain consistency maintenance
    - Passes reduced domains to next constraint (constraint propagation).

Early commercial successes

- Circuit design (Siemens)



- Container port scheduling (Hong Kong and Singapore)

- Real-time control (Siemens, Xerox)

Applications

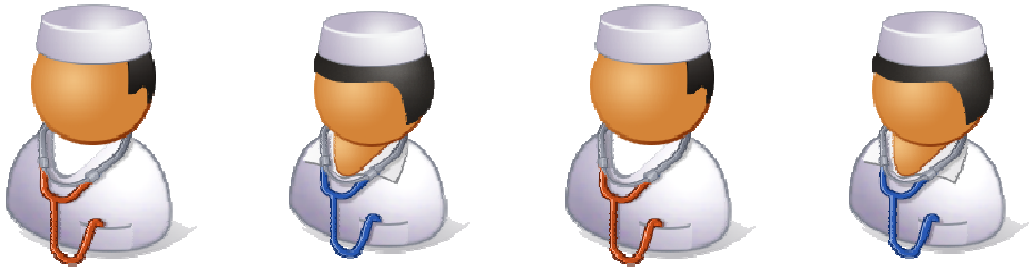
- Job shop scheduling
- Assembly line smoothing and balancing
- Cellular frequency assignment
- Nurse scheduling
- Shift planning
- Maintenance planning
- Airline crew rostering and scheduling
- Airport gate allocation and stand planning

Applications

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

Example: Employee scheduling

- Schedule four nurses in 8-hour shifts.
- A nurse works at most one shift a day, at least 5 days a week.
- Same schedule every week.
- No shift staffed by more than two different nurses in a week.
- A nurse cannot work different shifts on two consecutive days.
- A nurse who works shift 2 or 3 must do so at least two days in a row.



Two ways to view the problem

Assign nurses to shifts

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
Shift 1	A	B	A	A	A	A	A
Shift 2	C	C	C	B	B	B	B
Shift 3	D	D	D	D	C	C	D

Assign shifts to nurses

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
Nurse A	1	0	1	1	1	1	1
Nurse B	0	1	0	2	2	2	2
Nurse C	2	2	2	0	3	3	0
Nurse D	3	3	3	3	0	0	3

0 = day off

Use **both** formulations in the same model!

First, assign nurses to shifts.

Let w_{sd} = nurse assigned to shift s on day d

$$\text{alldiff}\,(w_{1d}, w_{2d}, w_{3d}),\quad \text{all } d$$

Schedule 3 different nurses on each day

Use **both** formulations in the same model!

First, assign nurses to shifts.

Let w_{sd} = nurse assigned to shift s on day d

$$\text{alldiff}\left(w_{1d}, w_{2d}, w_{3d}\right), \quad \text{all } d$$

$$\text{cardinality}\left(w \mid (A,B,C,D),(5,5,5,5),(6,6,6,6)\right)$$

Each nurse works at least 5 and at most 6 days a week

Use **both** formulations in the same model!

First, assign nurses to shifts.

Let w_{sd} = nurse assigned to shift s on day d

$$\text{alldiff}\left(w_{1d}, w_{2d}, w_{3d}\right), \quad \text{all } d$$

$$\text{cardinality}\left(w \mid (A,B,C,D),(5,5,5,5),(6,6,6,6)\right)$$

$$\text{nvalues}\left(w_{s,\text{Sun}},\ldots,w_{s,\text{Sat}} \mid 1,2\right), \quad \text{all } s$$

At least 1 and at most 2 nurses work any given shift.

Remaining constraints are not easily expressed in this notation.

So, assign shifts to nurses.

Let y_{id} = shift assigned to nurse i on day d

$$\text{alldiff}\left(y_{1d}, y_{2d}, y_{3d}\right), \text{ all } d$$

Assign a different nurse to each shift on each day.

Redundant, but speeds solution.

Remaining constraints are not easily expressed in this notation.

So, assign shifts to nurses.

Let y_{id} = shift assigned to nurse i on day d

$$\text{alldiff}\left(y_{1d}, y_{2d}, y_{3d}\right), \quad \text{all } d$$

$$\text{stretch}\left(y_{i,\text{Sun}}, \ldots, y_{i,\text{Sat}} \mid (2,3),(2,2),(6,6),P\right), \quad \text{all } i$$

Shift 2 or 3 must be worked at least 2 days in a row.

Remaining constraints are not easily expressed in this notation.

So, assign shifts to nurses.

Let y_{id} = shift assigned to nurse i on day d

$$\text{alldiff}\left(y_{1d}, y_{2d}, y_{3d}\right), \text{ all } d$$

$$\text{stretch}\left(y_{i,\text{Sun}}, \ldots, y_{i,\text{Sat}} \mid (2,3),(2,2),(6,6),P\right), \text{ all } i$$

Pattern constraint: $P = \{(s,0),(0,s) \mid s = 1,2,3\}$

Don't switch shifts without taking at least one day off.

Connect the w_{sd} variables to the y_{id} variables.

Use **channeling constraints**:

$$w_{y_{id}d} = i, \quad \text{all } i,d$$

$$y_{w_{sd}d} = s, \quad \text{all } s,d$$

Channeling constraints increase propagation and make the problem easier to solve.

The complete model is:

$$\text{alldiff}\left(w_{1d}, w_{2d}, w_{3d}\right), \quad \text{all } d$$

$$\text{cardinality}\left(w \mid (A,B,C,D),(5,5,5,5),(6,6,6,6)\right)$$

$$\text{nvalues}\left(w_{s,\text{Sun}},\ldots,w_{s,\text{Sat}} \mid 1,2\right), \quad \text{all } s$$

$$\text{alldiff}\left(y_{1d}, y_{2d}, y_{3d}\right), \quad \text{all } d$$

$$\text{stretch}\left(y_{i,\text{Sun}},\ldots,y_{i,\text{Sat}} \mid (2,3),(2,2),(6,6),P\right), \quad \text{all } i$$

$$w_{y_{id}d} = i, \quad \text{all } i,d$$

$$y_{w_{sd}d} = s, \quad \text{all } s,d$$

Example: Resource-Constrained Scheduling

- Use the **cumulative scheduling** constraint.
- Total resources consumed by jobs at any one time must not exceed L.

$$\text{cumulative}\left((t_1,\ldots,t_n),(p_1,\ldots,p_n),(c_1,\ldots,c_n),C\right)$$

Job start times (variables)

Job processing times

Job resource requirements

Ship loading

- The problem
  - Load 34 items on the ship in minimum time (min makespan)
  - Each item i requires p_i minutes and c_i workers.
  - Total of 8 workers available.
- From IBM OPL Studio manual

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

Use the cumulative scheduling constraint.

$$\begin{aligned}
\min \quad & z \\
\text{s.t.} \quad & z \geq t_i + p_i, \quad i = 1,\ldots,34 \\
& \text{cumulative}\big((t_1,\ldots,t_{34}),(p_1,\ldots,p_{34}),(c_1,\ldots,c_{34}),8\big) \\
& t_2 \geq t_1 + 3, \quad t_4 \geq t_1 + 3, \quad \text{etc. (precedence constraints)}
\end{aligned}$$

Note that there are no integer variables.

The solver may treat the t_i s as integers, but this is an implementation decision.

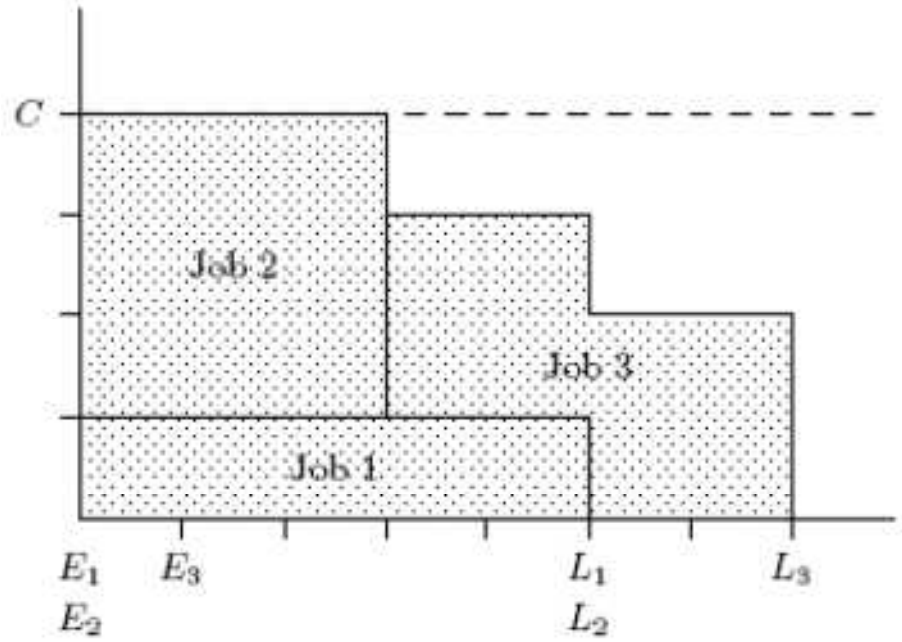
Edge finding for cumulative scheduling

Consider a cumulative scheduling constraint:

$$\text{cumulative}\big((s_1,s_2,s_3),(p_1,p_2,p_3),(c_1,c_2,c_3),C\big)$$

j	p_j	c_j	E_j	L_j
1	5	1	0	5
2	3	3	0	5
3	4	2	1	7

A feasible solution:

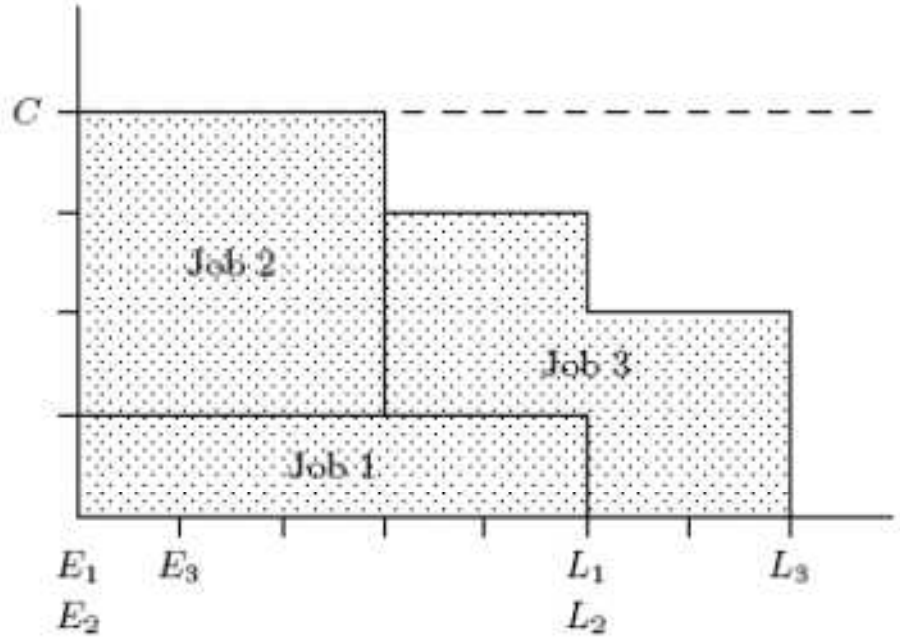


Edge finding for cumulative scheduling

We can deduce that job 3 must finish after the others finish: $3 > \{1,2\}$

Because the total **energy** required exceeds the area between the earliest release time and the later deadline of jobs 1,2:

$$e_3 + e_{\{1,2\}} > C \cdot \left(L_{\{1,2\}} - E_{\{1,2,3\}} \right)$$



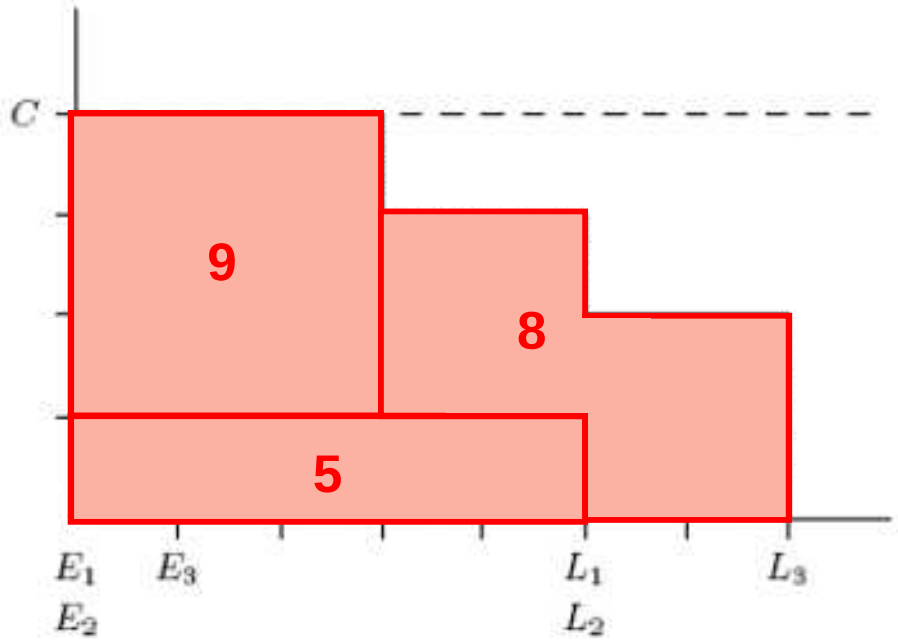
Edge finding for cumulative scheduling

We can deduce that job 3 must finish after the others finish: $3 > \{1,2\}$

Because the total **energy** required exceeds the area between the earliest release time and the later deadline of jobs 1,2:

$$e_3 + e_{\{1,2\}} > C \cdot \left(L_{\{1,2\}} - E_{\{1,2,3\}} \right)$$

Total energy required = 22



Edge finding for cumulative scheduling

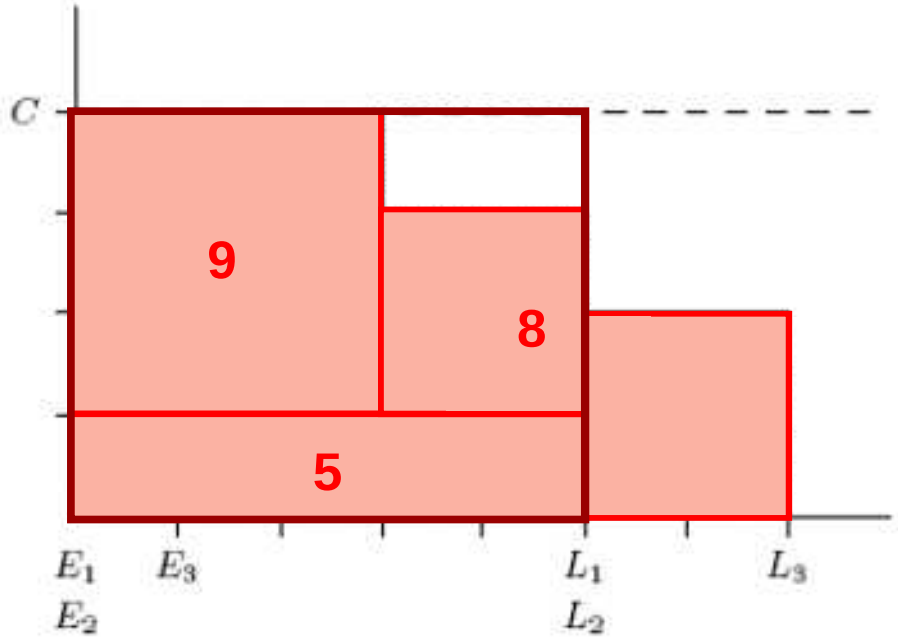
We can deduce that job 3 must finish after the others finish: $3 > \{1,2\}$

Because the total **energy** required exceeds the area between the earliest release time and the later deadline of jobs 1,2:

$$e_3 + e_{\{1,2\}} > C \cdot \left(L_{\{1,2\}} - E_{\{1,2,3\}} \right)$$

Total energy required = 22

Area available = 20



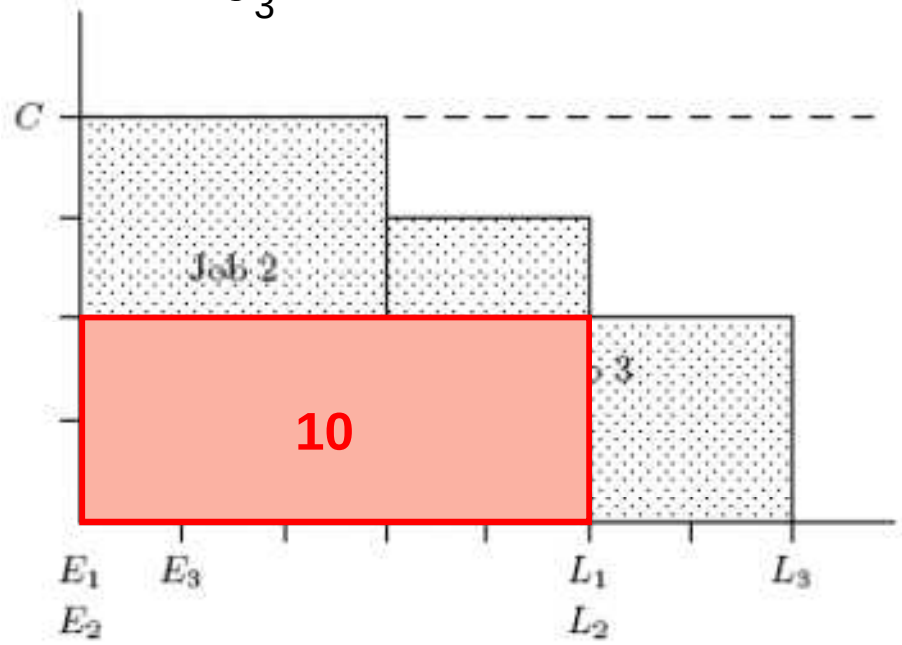
Edge finding for cumulative scheduling

We can deduce that job 3 must finish after the others finish: $3 > \{1,2\}$

We can update the release time of job 3 to

$$E_{\{1,2\}} + \frac{e_J - (C - c_3)(L_{\{1,2\}} - E_{\{1,2\}})}{c_3}$$

Energy available for jobs 1,2 if space is left for job 3 to start anytime = 10



Edge finding for cumulative scheduling

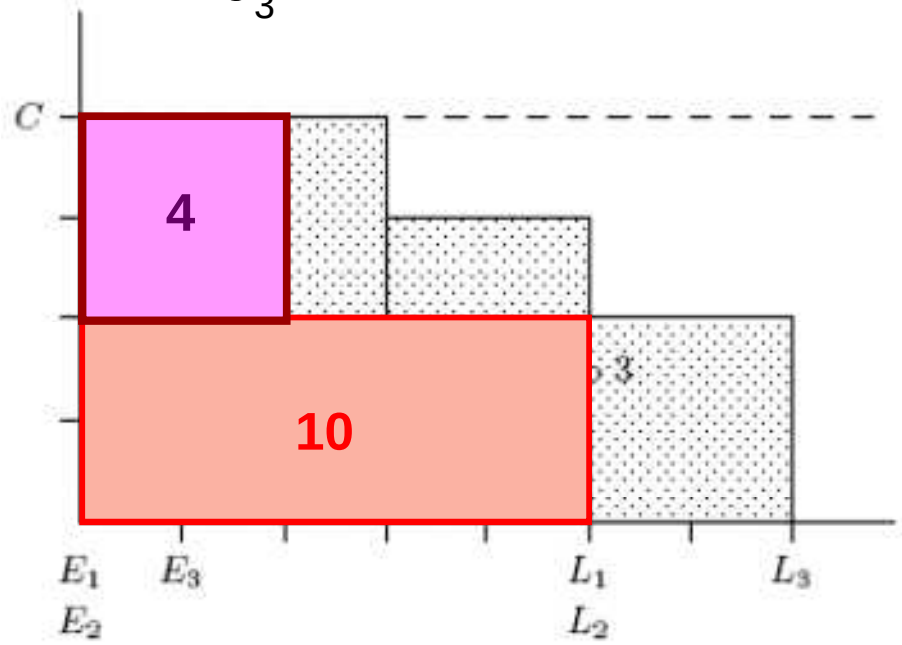
We can deduce that job 3 must finish after the others finish: $3 > \{1,2\}$

We can update the release time of job 3 to

$$E_{\{1,2\}} + \frac{e_J - (C - c_3)(L_{\{1,2\}} - E_{\{1,2\}})}{c_3}$$

Energy available for jobs 1,2 if space is left for job 3 to start anytime = 10

Excess energy required by jobs 1,2 = 4



Edge finding for cumulative scheduling

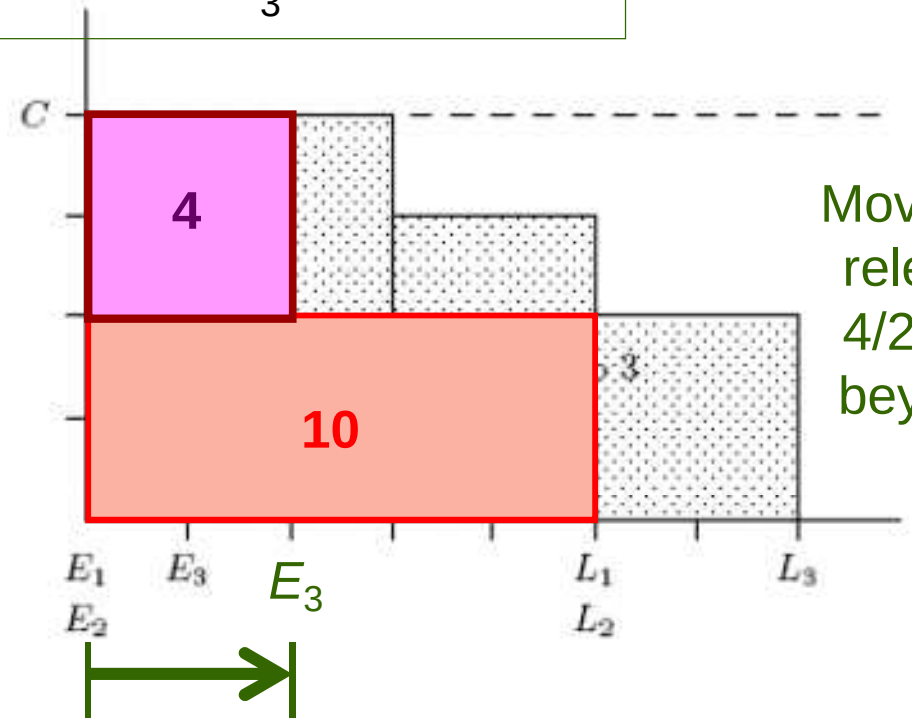
We can deduce that job 3 must finish after the others finish: $3 > \{1,2\}$

We can update the release time of job 3 to

$$E_{\{1,2\}} + \frac{e_J - (C - c_3)(L_{\{1,2\}} - E_{\{1,2\}})}{c_3}$$

Energy available for jobs 1,2 if space is left for job 3 to start anytime = 10

Excess energy required by jobs 1,2 = 4



Move up job 3 release time 4/2 = 2 units beyond $E_{\{1,2\}}$

Edge finding for cumulative scheduling

In general, if $e_{J\cup\{k\}} > C \cdot \left(L_J - E_{J\cup\{k\}}\right)$

then $k > J$, and update E_k to

$$\max_{\substack{J'\subset J \\ e_{J'}-(C-c_k)(L_{J'}-E_{J'})>0}} \left\{ E_{J'} + \frac{e_{J'} - (C-c_k)(L_{J'}-E_{J'})}{c_k} \right\}$$

In general, if $e_{J\cup\{k\}} > C \cdot \left(L_{J\cup\{k\}} - E_J\right)$

then $k < J$, and update L_k to

$$\min_{\substack{J'\subset J \\ e_{J'}-(C-c_k)(L_{J'}-E_{J'})>0}} \left\{ L_{J'} - \frac{e_{J'} - (C-c_k)(L_{J'}-E_{J'})}{c_k} \right\}$$

Edge finding for cumulative scheduling

There is an $O(n^2)$ algorithm that finds all applications of the edge finding rules.

Other propagation rules for cumulative scheduling

- Extended edge finding.
- Timetabling.
- Not-first/not-last rules.
- Energetic reasoning.

CP and Mathematical Programming

Comparison

CP	MP
Logic processing	Numerical calculation
Inference (filtering, constraint propagation)	Relaxation
High-level modeling (global constraints)	Atomistic modeling (linear inequalities)
Branching	Branching
Constraint-based processing	“Independence” of model and algorithm

Software for Constraint Programming

- ECLiPSe (NICTA), open source
  - Early CP (and hybrid) solver, still maintained
- CHIP (Cosytec), commercial
  - State-of-the-art solver
- OPL CP Optimizer (IBM), commercial
  - State-of-the-art solver, originally developed by ILOG
- Gecode (Schulte & Tack), free download
  - State-of-the-art toolkit for building CP solvers
- Frontline MIP/CP solver (Frontline Systems), commercial
  - Add-in for Excel spreadsheets
- G12 (NICTA), under development
  - Major CP and hybrid system
- Google OR-tools (Google), open source
  - Includes CP solver

Integrating OR and CP

Complementary strengths

How to integrate

Constraint propagation + relaxation

CP-based branch and price

Decomposition methods

Complementary Strengths

- CP:
  - Inference methods
  - Modeling
  - Exploits local structure
  - Good at scheduling
- OR:
  - Relaxation methods
  - Duality theory
  - More robust
  - Good with continuous variables

Let's bring them together!

How to Integrate

- Constraint propagation + relaxation
  - Propagation reduces search space.
  - Relaxation bounds prune the search
- CP-based column generation
  - In branch-and-price methods
  - CP accommodates complex constraints on columns
- Decomposition methods
  - Distinguish master problem and subproblem
  - MILP solves one, CP the other.
- Use CP-style modeling

Constraint propagation + relaxation

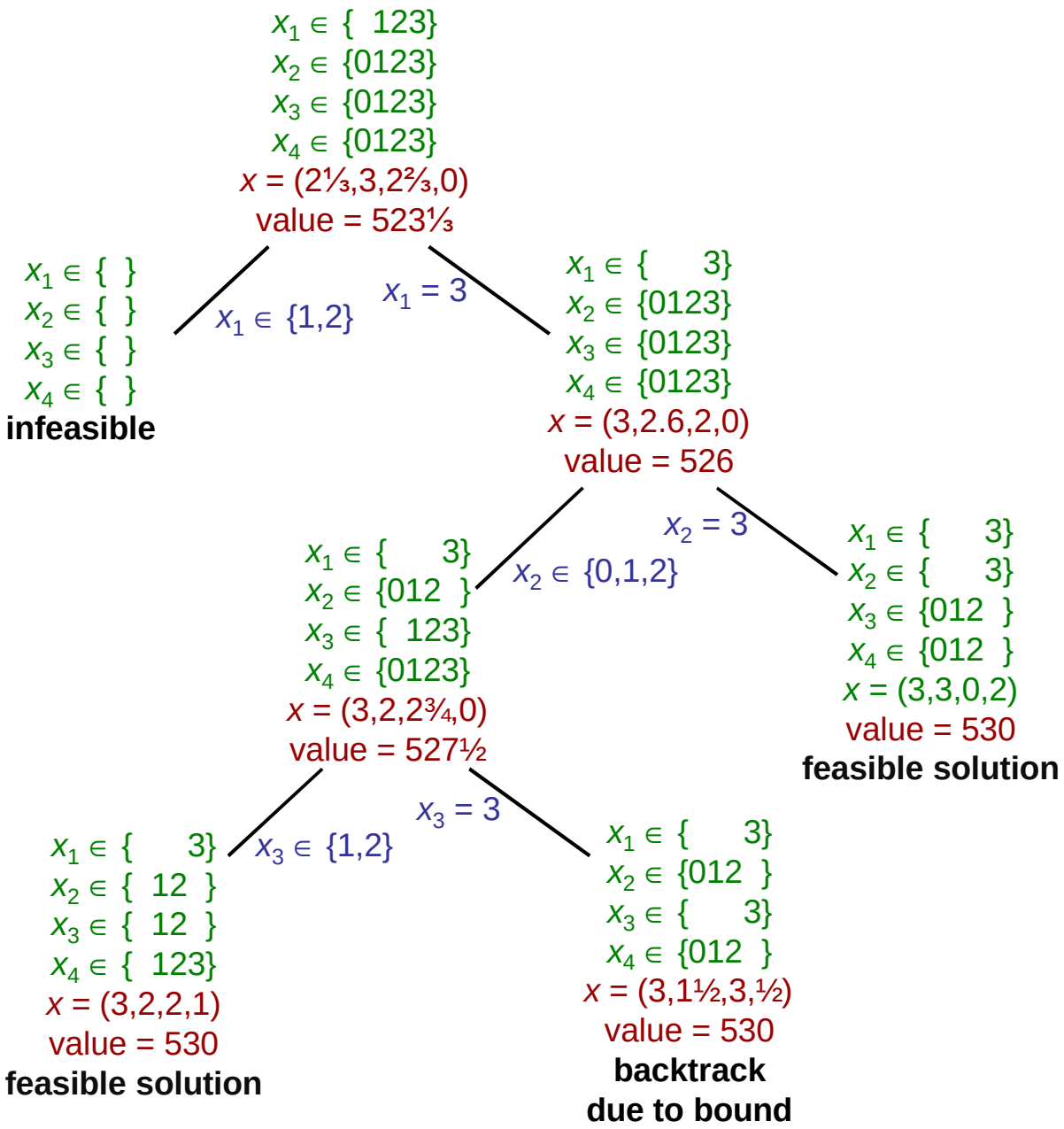
- Combine MIP-style bounding with CP-style propagation.
- At each node of search tree:
  - Solve linear relaxation to get bound, as in MIP.
  - Filter variable domains and propagate constraints, as in CP.

Search tree for an IP problem

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

$$x_3 + x_4 \geq 2$$
$$x_2 + x_4 \geq 2$$
$$x_2 + x_3 \geq 3$$

Knapsack cuts

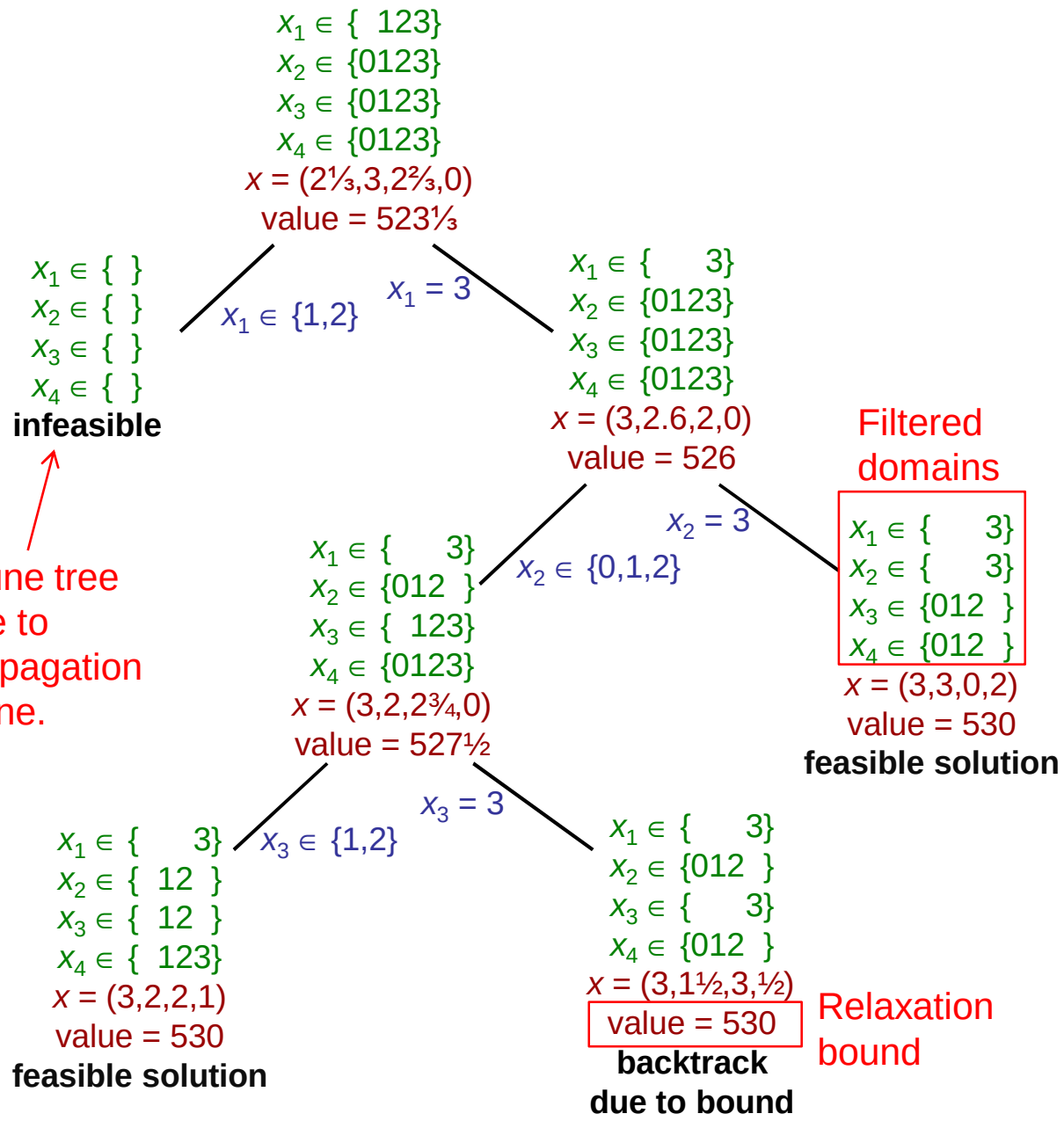


Search tree for an IP problem

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

$$x_3 + x_4 \geq 2$$
$$x_2 + x_4 \geq 2$$
$$x_2 + x_3 \geq 3$$

Knapsack cuts



Constraint propagation + relaxation

- **Example: Relaxation of cumulative constraint**
  - Use discrete-time MILP model

$$z_t = z_{t-1} + \sum_j c_j x_{jt} - \sum_j c_j x_{j,t-p_j}, \quad \text{all } t$$

$$\sum_t x_{jt} = 1, \quad \text{all } j$$

$$z_t \le C, \quad \text{all } t$$

$$0 \le x_{jt} \le 1, \quad \text{all } j,t$$

Total resource consumption at time *t*

= 1 when job j starts at time t

  - Add this to linear relaxation of problem while propagating the cumulative constraint.
    - 0-1 variables x_{jt} appear only in the relaxation

Constraint propagation + relaxation

- **Example: Relaxation of cumulative constraint**
  - Or use valid inequalities in original variables t_j

$$\sum_{j \in J} t_j \geq k r_J + \frac{1}{C} \sum_{i=1}^{k} (k - i + 1) p_{j_i} c_{j_i} - \sum_{i=1}^{k} p_{j_i}$$

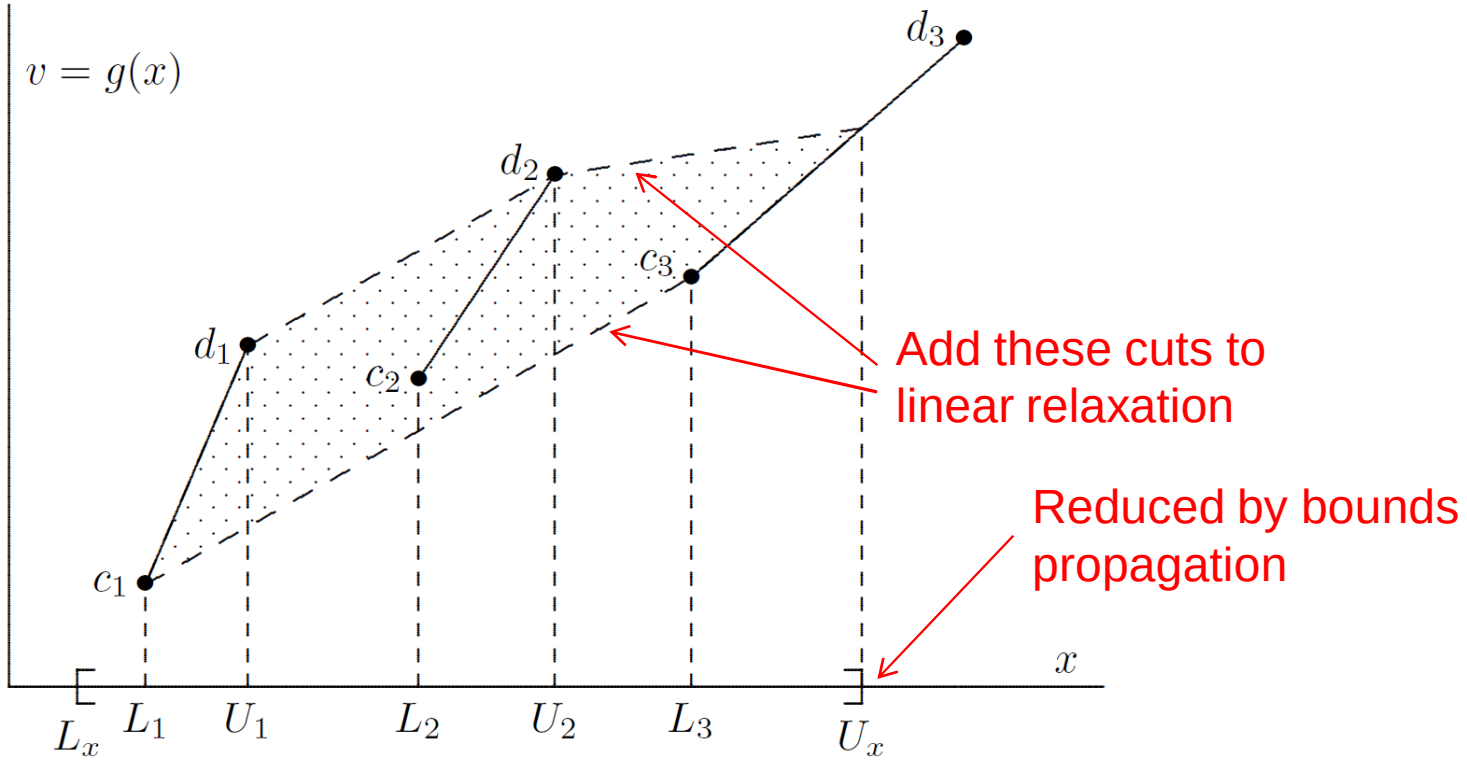
Earliest release time in set J

$$\sum_{j \in J} t_j \leq k r_J - \frac{1}{C} \sum_{i=1}^{k} (k - i + 1) p_{j_i} c_{j_i}$$

where $J = \{i_1, \ldots, i_k\}$ is any subset of jobs.

Constraint propagation + relaxation

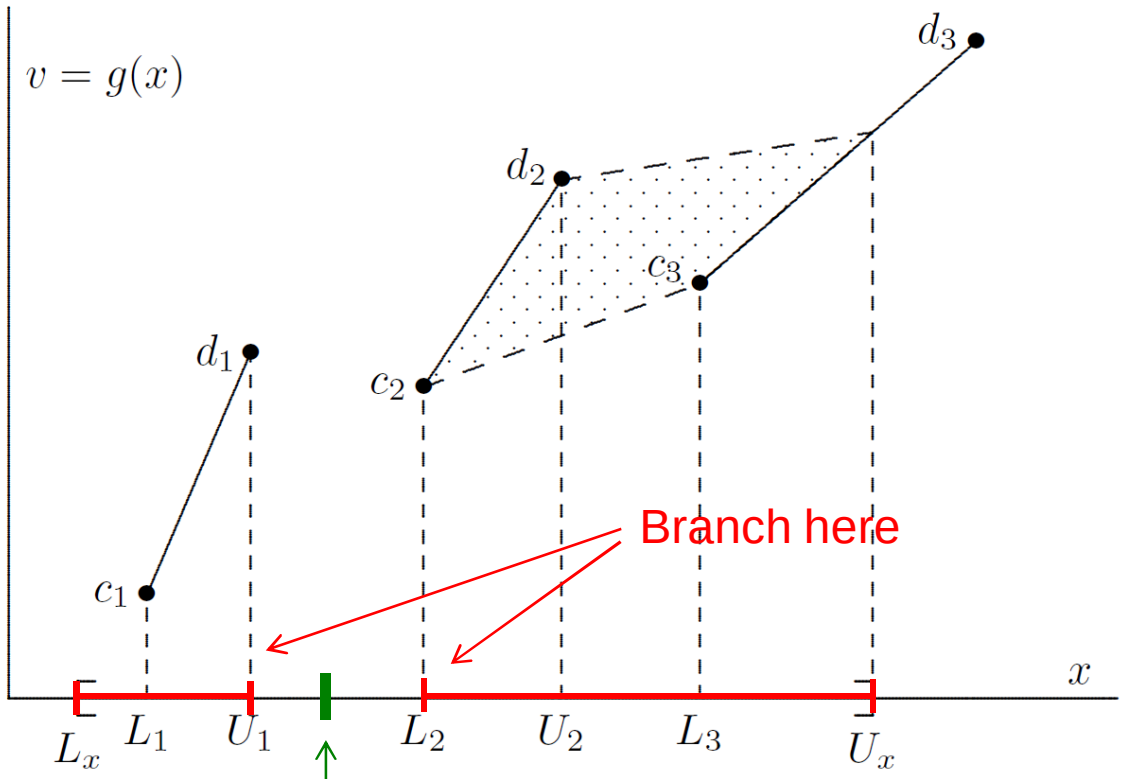
- **Example: Piecewise linear functions**
  - Use convex hull relaxation in original variables.



  - No need for 0-1 variables

Constraint propagation + relaxation

- Example: Piecewise linear functions
  - Branch on original variables & tighten relaxation.



Computational Advantage of Integrating CP and OR

Using CP + relaxation from OR

Problem	*Speedup*	
Lesson timetabling (LP + red. cost var. fixing)	2 to 50 times faster than CP	Focacci, Lodi, Milano (1999)
Piecewise linear costs (LP relaxation)	2 to 120 times faster than MILP	Refalo (1999), Yunes, Aron & Hooker (2010)
Flow shop scheduling, etc. (LP relaxation + cuts)	4 to 150 times faster than MILP.	Hooker & Osorio (1999)

Computational Advantage of Integrating CP and OR

Using CP + relaxation from OR

Problem	*Speedup*	
Product configuration (LP relaxation + cuts)	30 to 40 times faster than CP, MILP	Thorsteinsson & Ottosson (2001)
Automatic recording (Lagrangean relaxation)	1 to 10 times faster than CP, MILP	Sellmann & Fahle (2001)
Stable set problem (semidefinite relaxation)	Better than CP in less time	Van Hoeve (2001)

Computational Advantage of Integrating CP and OR

Using CP + relaxation from OR

Problem	*Speedup*	
Structural design (nonlinear) (LP quasi-relaxation + logic cuts)	Up to 600 times faster than MILP.	Bollapragada, Ghattas & Hooker (2001)
Radiation therapy planning (Lagrangean relaxation)	10 times faster than CP, MILP	Cambazard, O'Mahony and O'Sullivan (2010)

Applications of Integrated CP and OR

Using CP + relaxation from OR

Application	
Orthogonal Latin squares	Appa, Magos & Mourtos (2002)
Truss structure design	Bollapragada, Gattas & Hooker (2001)
Chemical processing network design	Grossmann et al. (1994), Hooker & Osorio (1999)
Vehicle routing	Sadykov (2004)
Resource-constrained scheduling	Demassy, Artiques & Michelon (2002), Berthold et al. (2010)
Multiple machine scheduling	Bockmayr & Pisaruk (2003)
Shuttle transit routing	Quadrifoglio, Dessouky & Ordóñez (2008)
Boat party scheduling	Hooker & Osorio (1999)

Applications of Integrated CP and OR

Using CP + relaxation from OR

Application	
Multidimensional knapsack problem	Osorio & Glover (2001)
Factory retrofit planning	Sawaya & Grossmann (2005)
Strip packing	Sawaya & Grossmann (2005)
Chemical process engineering	Lee & Grossmann (2003), Vecchietti et al (2001), Swaya & Grossmann (2007)
Transport & production problems with piecewise linear costs	Refalo (1999), Ottosson et al. (1999)
TSP with time windows	Milano & van Hoeve (2002)
Product configuration	Milano & van Hoeve (2002)

Applications of Integrated CP and OR

Using CP + relaxation from OR

Application	
Network design	Cronholm & Ajili (2004)
Automatic digital recording	Sellmann & Fahle (2001)
Traveling tournament problem	Benoist, Laburthe & Rottembourg (2001)
Resource-constrained shortest path problem	Gellermann, Sellmann & Wright (2005)
Radiation therapy planning	Cambazard, O'Mahony & O'Sullivan (2010)
CP domain filtering	Khemmoudj, Bennaceur & Nagih (2005)

CP-Based Branch and Price

- Same as traditional branch and price…
  - Except that CP generates the columns.
- Introduce an integer variable for each combinatorial possibility.
  - Removes most or all of symmetry.
  - Filter variable domains and propagate constraints, as in CP.
- At each node of MIP search tree:
  - Solve linear relaxation using column generation.
  - Generate improving columns with CP (pricing problem).
  - Start with these columns at child nodes.

Airline Crew Scheduling

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	12	14

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.

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

$$\min\ z$$

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

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

= 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

$$\min\ z$$

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

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

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

$$\min\ z$$

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

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

= 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

$$\min\ z$$

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

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

Airline Crew Scheduling

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

$$\min\ z$$

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

Dual variables

(10)	u_1
(9)	u_2
(0)	v_1
(0)	v_2
(0)	v_3
(0)	v_4
(0)	v_5
(3)	v_6

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

Airline Crew Scheduling

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

$$\min\ z$$

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

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

Dual variables

(10)	u_1
(9)	u_2
(0)	v_1
(0)	v_2
(0)	v_3
(0)	v_4
(0)	v_5
(3)	v_6

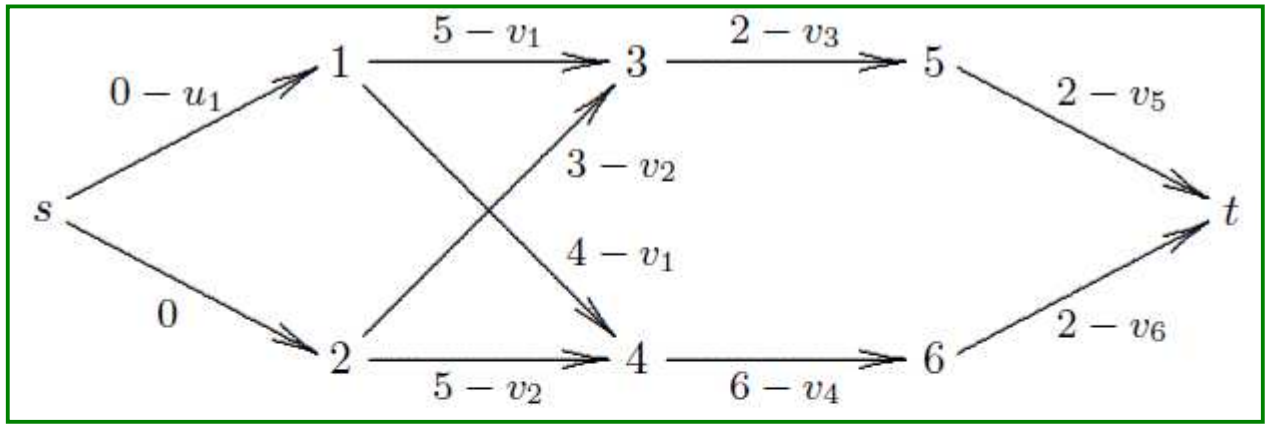
The reduced cost of an excluded roster *k* for crew member *i* is

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

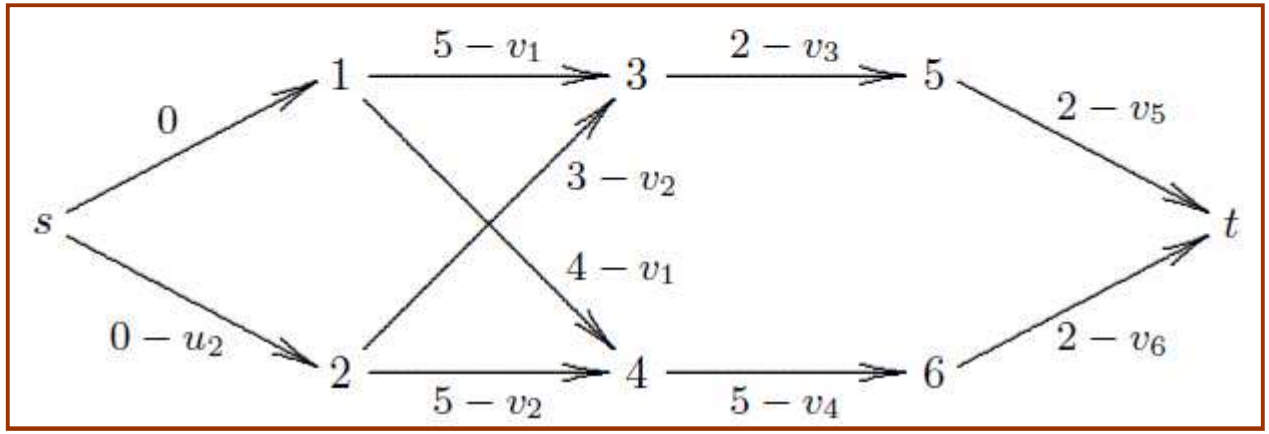
We will formulate the pricing problem as a shortest path problem.

Pricing problem

Crew member 1



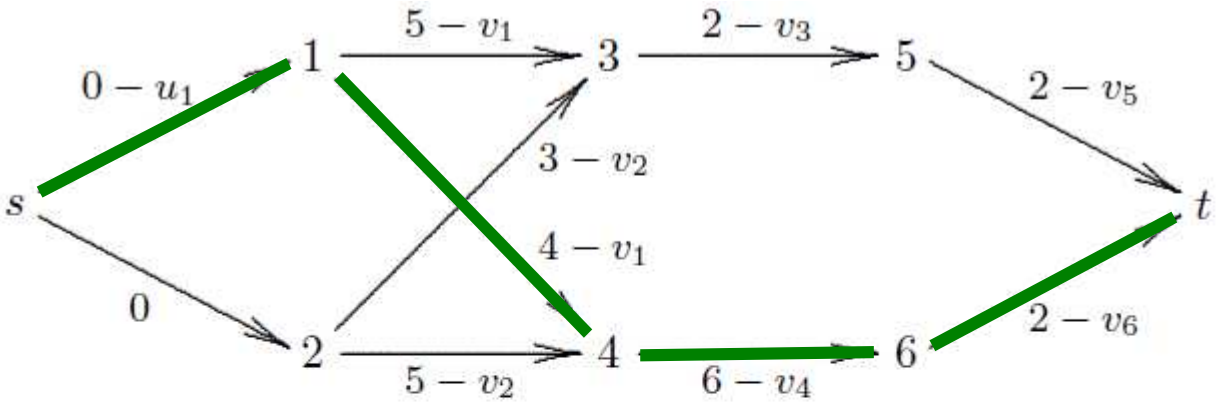
Crew member 2



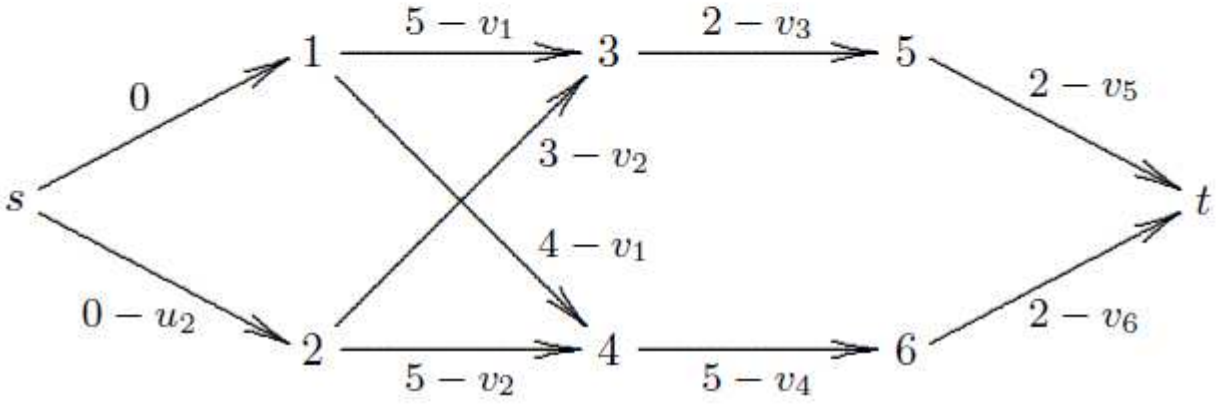
Pricing problem

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

Crew member 1



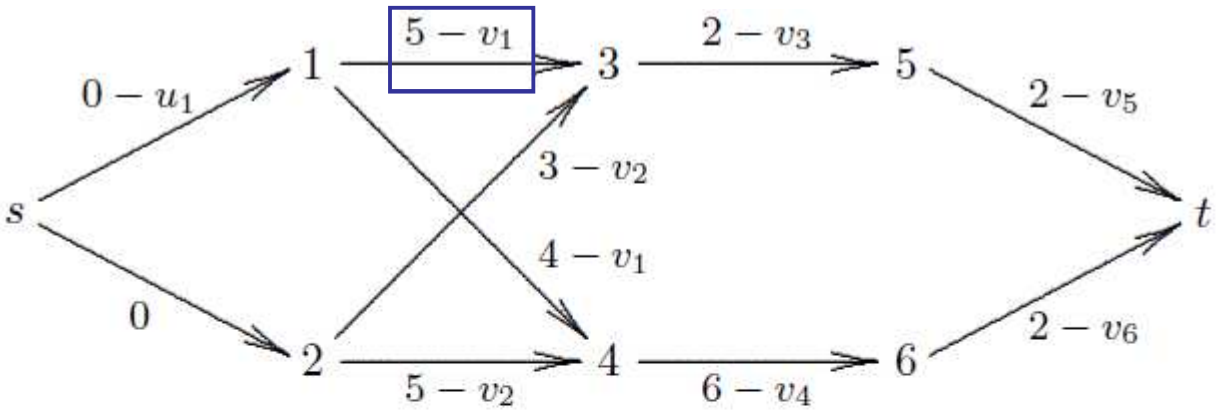
Crew member 2



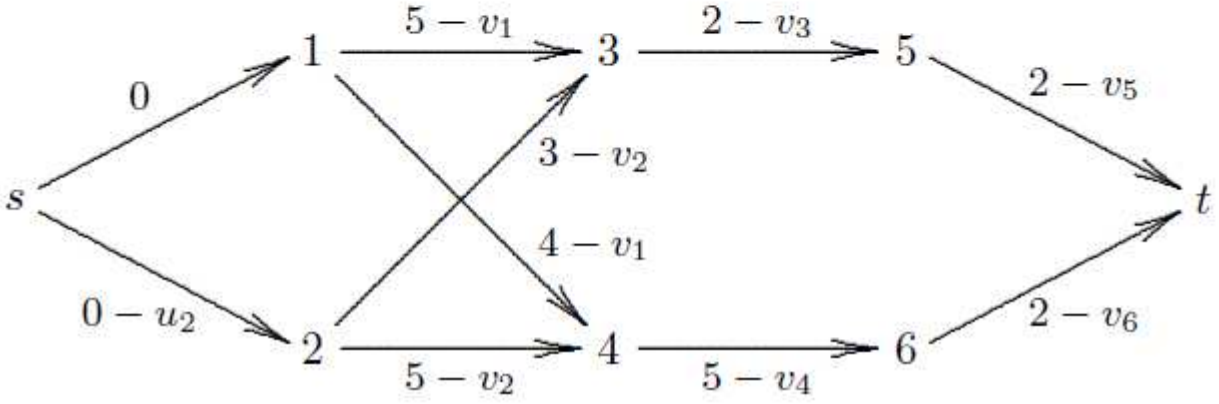
Pricing problem

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

Crew member 1



Crew member 2



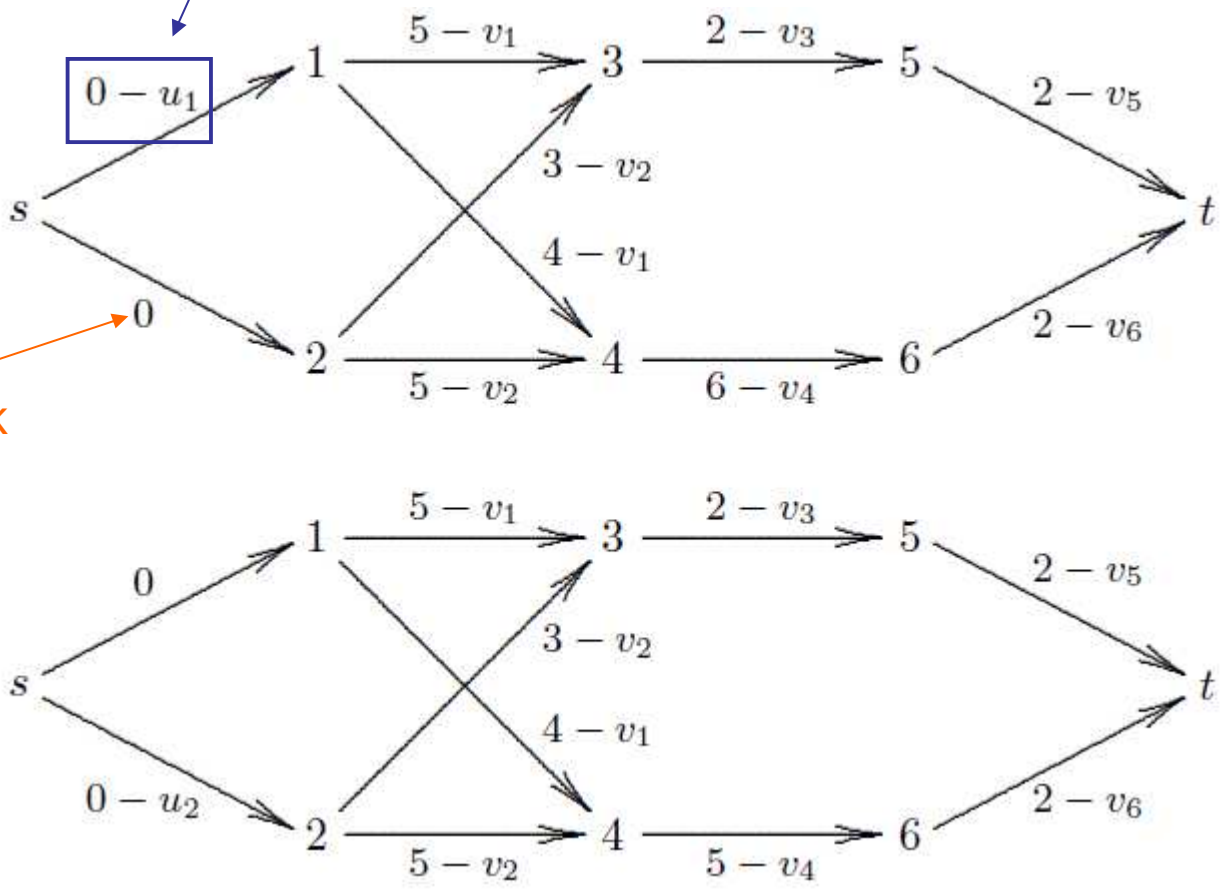
Pricing problem

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

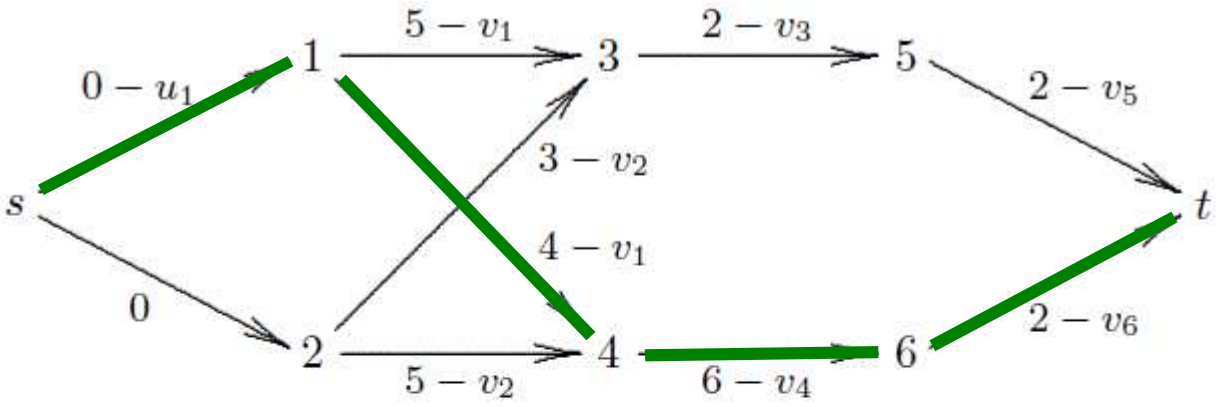
Crew member 2



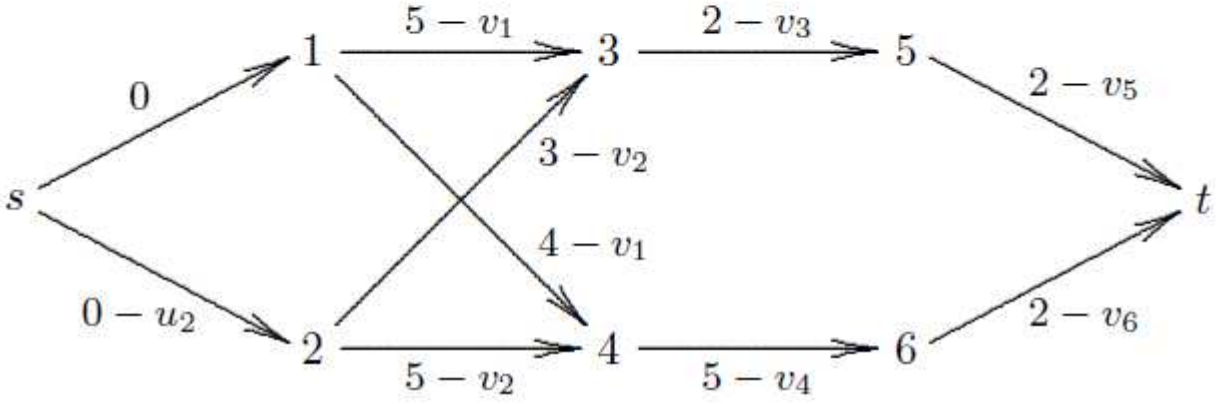
Pricing problem

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

Crew member 1



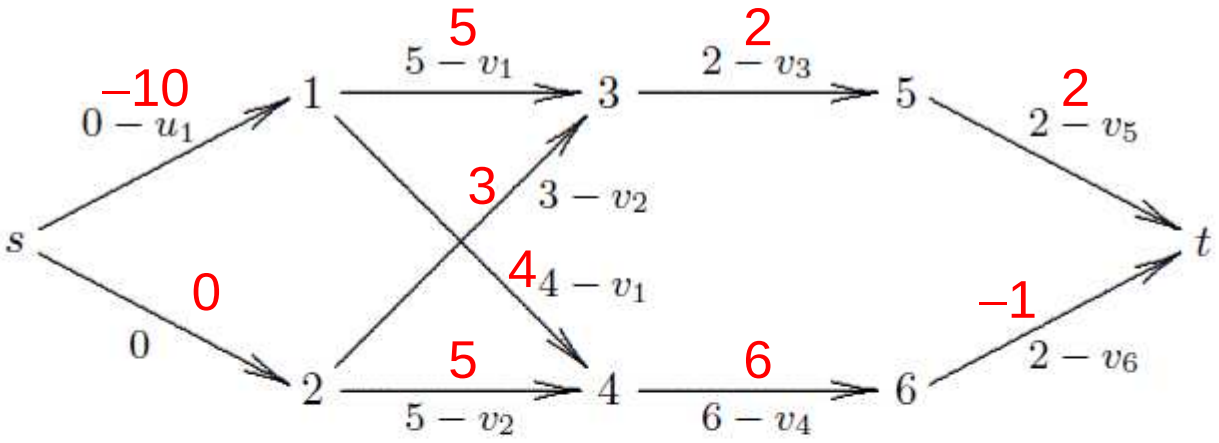
Crew member 2



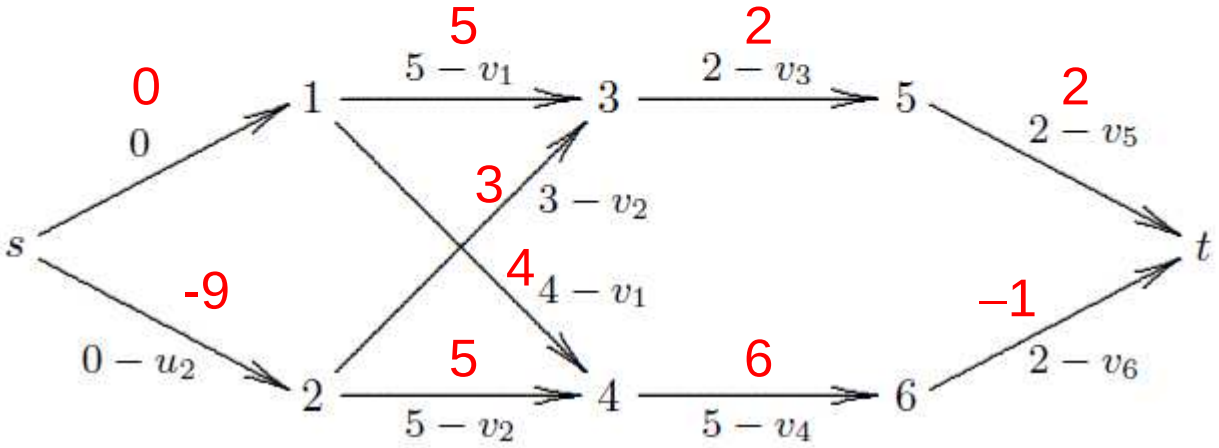
Pricing problem

Arc lengths using dual solution of LP relaxation

Crew member 1



Crew member 2

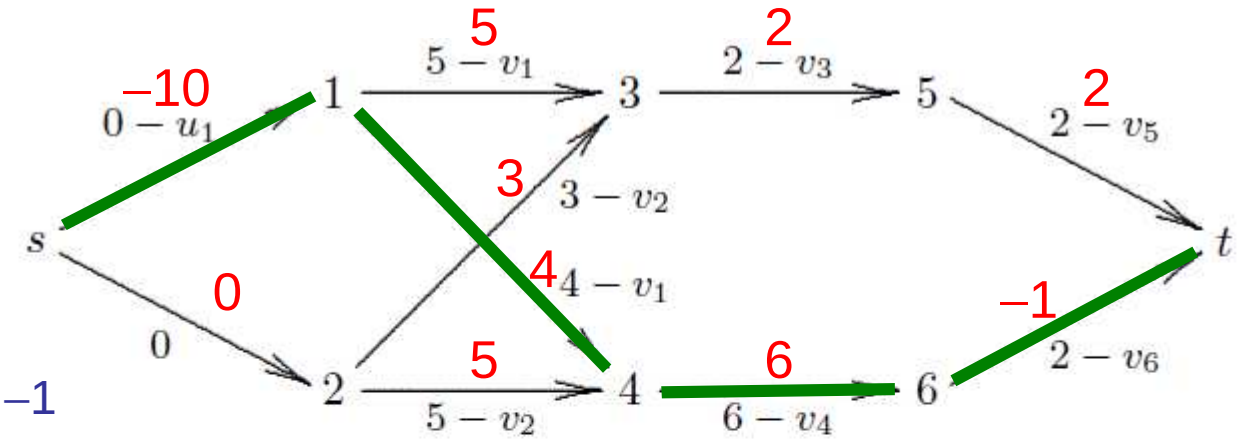


Pricing problem

Solution of shortest path problems

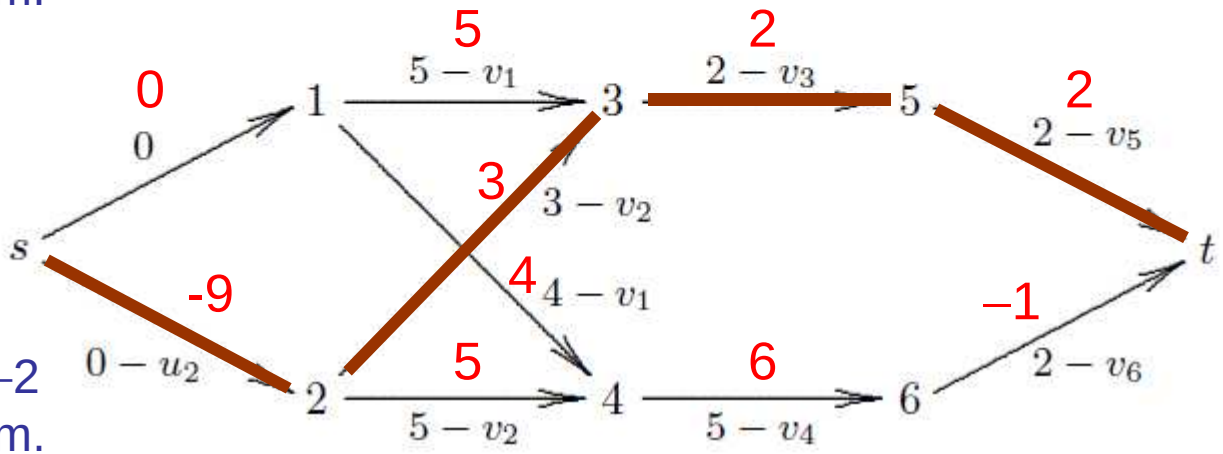
Crew member 1

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 duration of flights.

It **can** be solved by CP:

Set of flights assigned to crew member *i*

Path length

Graph

Path global constraint → $\text{Path}(X_i, z_i, G),\ \text{all flights } i$

Setsum global constraint → $T_{\min} \le \sum_{j \in X_i} (f_j - s_j) \le T_{\max}$

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

Duration of flight *j*

Process Scheduling

Assign batches to processing-unit/start-time combinations while observing time constraints.

A **schedule** is the sequence of unit/start-time combinations assigned to a batch.

The time gap between two consecutive processes in a schedule is bounded.

Total time for a schedule is bounded.

Process Scheduling

The integer programming model is

$$\min \sum_{ij} c_{ij} x_{ij}$$

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

$$\sum_{ij} a_{jk} x_{ij} \le 1, \quad \text{all } k \quad (v_k)$$

$$x_{ij} \in \{0,1\}, \quad \text{all } i, j$$

Cost of assigning batch *i* to schedule *j*

= 1 if we assign batch i to schedule j

Each batch is assigned to exactly 1 schedule.

= 1 if schedule *j* includes unit/start-time *k*

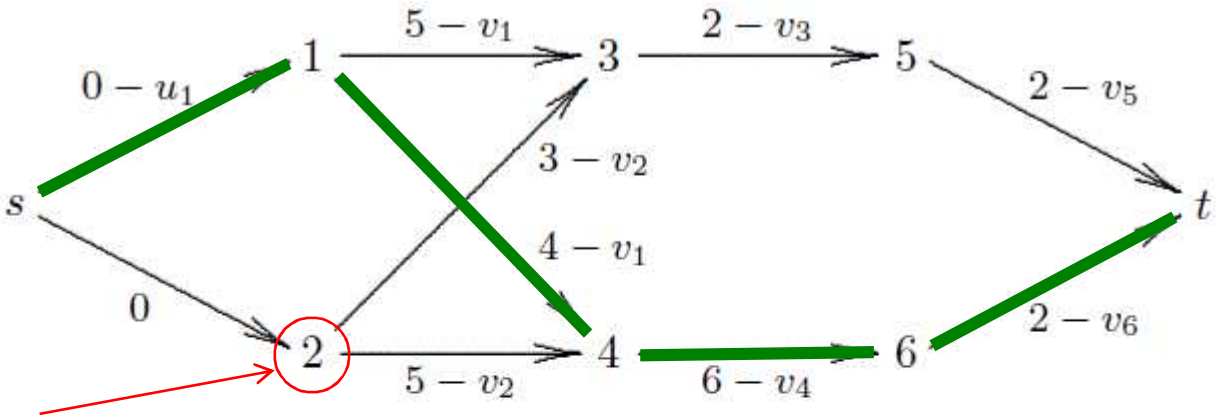
Each unit/start-time is assigned at most one batch.

Pricing problem

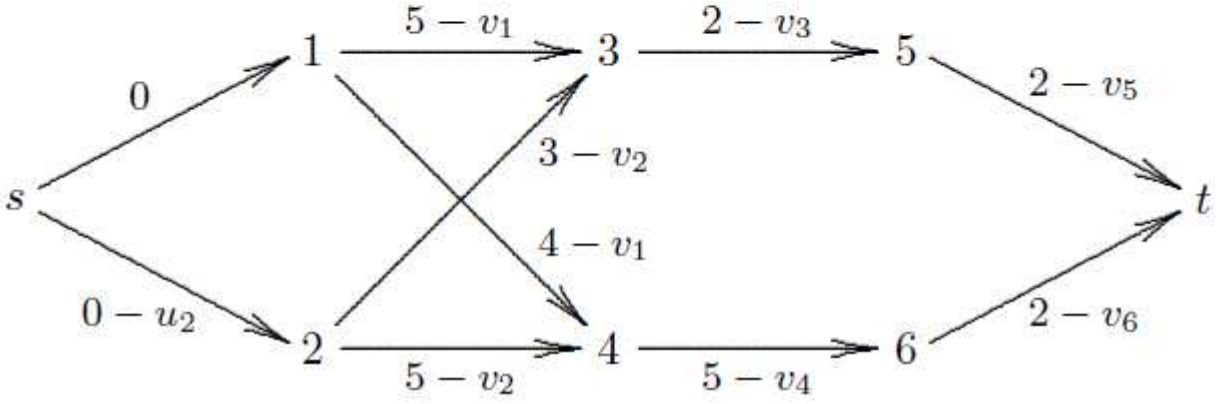
Each s-t path corresponds to a schedule, provided the total processing time is within bounds.

Batch 1

Unit/start-time combination



Batch 2



Everything else is the same as before, except network generation.

Computational Advantage of Integrating CP and OR

Using CP-based Branch and Price

Problem	*Speedup*	
Urban transit crew scheduling	Schedules 210 trips, vs. 120 for traditional branch and price	Yunes, Moura & de Souza (1999)
Airline crew rostering	Incorporates complicated work rules	Fahle et al. (2002)
Traveling tournament scheduling	First to solve 8-team instance	Easton, Nemhauser & Trick (2002)

Applications of Integrated CP and OR

Using CP-based branch and price

Application	
Airline crew rostering	Junker et al. (1999), Kohl (2000), Chabrier (2000), Fahle et al. (2002), Sellmann et al. (2002)
Aircraft scheduling	Grönqvist (2003)
Bus crew scheduling	Yunes, Moura & de Souza (2005)
Vehicle routing	Rousseau, Gendreau & Pesant (2002)
Network design	Chabrier (2003)
Employee timetabling	Demassey, Pesant & Rousseau (2005)
Physician scheduling	Gendron, Lebbah & Pesant (2005)
Radiation therapy planning	Cambazard, O'Mahony & O'Sullivan (2010)
Traveling tournament problem	Easton, Nemhauser & Trick (2002), Trick & Yildiz (2007)

Decomposition Methods

- Can be applied to planning and scheduling problems.
  - MILP solves planning (master) problem.
  - CP solves scheduling subproblem.
- Link master and subproblem with logic-based Benders cuts.

Facility assignment and scheduling

Assign jobs to facilities, and schedule them, to minimize makespan.

Master problem assigns jobs, using MILP.

Subproblem decomposes into a separate scheduling problem for each facility, solved by CP.

$$\min \ M$$

$$M \geq t_j + p_{x_j j}, \ \text{all } j$$

Start time of job j

$$r_j \leq t_j \leq d_j - p_{x_j j}, \ \text{all } j$$

$$\text{cumulative}\left((t_j \,|\, x_j = i), (p_{ij} \,|\, x_j = i), (c_{ij} \,|\, x_j = i)\right), \ \text{all } i$$

Facility assigned to job j

Facility assignment and scheduling

Subproblem for each facility *i*, given an assignment *x* from master

$$\min\ M$$

$$M \geq t_j + p_{x_j j},\ \text{all } j$$

$$r_j \leq t_j \leq d_j - p_{x_j j},\ \text{all } j$$

$$\text{cumulative}\left((t_j \,|\, x_j = i),(p_{ij} \,|\, x_j = i),(c_{ij} \,|\, x_j = i)\right)$$

Sample Benders cut (all release times the same):

$$M \geq M_{ik}\left(\sum_{j \in J_{ik}} p_{ij}(1 - y_{ij}) + \max_{j \in J_{ik}}\{d_j\} - \min_{j \in J_{ik}}\{d_j\}\right)$$

Deadline for job *j*

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

=1 if job *j* assigned to facility *i* ($x_j = i$)

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

Facility assignment and scheduling

The master problem is

$$\min M$$

$$M \geq M_{ik}\left(\sum_{j \in J_{ik}} p_{ij}(1-y_{ij}) + \max_{j \in J_{ik}}\{d_j\} - \min_{j \in J_{ik}}\{d_j\} \right), \quad \text{all i,}k$$

Relaxation of subproblem

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

Benders cuts

Computational Advantage of Integrating CP and OR

Using CP/MILP Benders methods

Problem	*Speedup*	
Min-cost planning & scheduing	20 to 1000 times faster than CP, MILP	Jain & Grossmann (2001), Thorsteinsson (2001)
Min-cost planning & scheduling	Solved some MIP-intractable instances in <1 sec	Yunes, Aron & Hooker (2010)
Polypropylene batch scheduling at BASF	Solved previously insoluble problem in 10 min	Timpe (2002)

Computational Advantage of Integrating CP and OR

Using CP/MILP Benders methods

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

Computational Advantage of Integrating CP and OR

Using CP/MILP Benders methods

Problem	*Speedup*	
Sports scheduling	Several orders of magnitude speedup vs MILP	Rasmussen & Trick (2007)
Single-facility scheduling	Schedules several times as many jobs as MILP. Faster and/or more robust than CP.	Coban & Hooker (2012)

Applications of Integrated CP and OR

Using CP/MILP Benders methods

Application	
Logic circuit verification	Hooker & Yan (1995)
Propositional satisfiability	Hooker (2000), Hooker & Ottosson (2003)
Planning & scheduling	Jain & Grossmann (2001), Hooker (2000,2004,2005), Harjunkoski & Grossmann (2002), Chu & Xia (2005)
Dispatch of automated vehicles	Corréa, Langevin & Rousseau (2004)
Steel production scheduling	Harjunkoski & Grossmann (2001)
Chemical batch scheduling	Timpe (2002), Maravelias & Grossmann (2004)
Computer processor scheduling	Cambazard et al. (2004), Benini et al. (2005,2008),

Applications of Integrated CP and OR

Using CP/MILP Benders methods

Application	
Location-allocation	Fazel-Zarandi & Beck (2009)
Traffic diversion	Xia, Eremin & Wallace (2004)
Transport network design	Peterson & Trick (2009)
Queuing design & control	Terekhov, Beck & Brown (2007)
Sports scheduling	Rasmussen & Trick (2007), Cheung (2009)

Software for Integrated Methods

- ECLiPSe (NICTA), open source
  - Exchanges information between ECLiPSEe solver, Xpress-MP
- IBM OPL Studio (IBM), commercial
  - Combines CPLEX and ILOG CP Optimizer with script language
- Mosel (FICO), commercial
  - Combines Xpress-MP, Xpress-Kalis with low-level modeling
- SIMPL (CMU), free download
  - Full integration with high-level modeling (prototype)
- SCIP (ZIB), free download, commercial users asked to buy license
  - Combines MILP and CP-based propagation
- G12 (NICTA), under development
  - Converts generic model to one that invokes cooperating solvers
- BARON (global optimization), commercial
  - Combines convexification and interval propagation

Some Very Recent Work

Benders for scheduling

Cutting planes from CP model

BDDs as constraint store

BDDs for relaxation bounds

Recent work – Benders for Scheduling

Joint work with Elvin Coban.

Apply logic-based Benders to single-facility scheduling with long time horizons and many jobs.

Decompose the problem by assigning jobs to segments of time horizon.

Segmented problem – Jobs cannot cross segment boundaries (e.g., weekends).

Unsegmented problem – Jobs can cross segment boundaries.

Recent work – Benders for Scheduling

Segmented instances, tight time windows

Table 4: Computation times in seconds for the segmented problem with tight time windows. The number of segments is 10% the number of jobs. Ten instances of each size are solved.

	Feasibility			Makespan			Tardiness		
Jobs	CP	MILP	Bndrs	CP	MILP	Bndrs	CP	MILP	Bndrs
60	0.1	14	1.9	60	7.7	6.4	0.1	16	3.0
80	181*	45	2.7	420*	147	11	63*	471*	20
100	199*	58	4.3	600*	600	17	547*	177*	11
120	272*	137	4.8	600*	600	39	600*	217*	2.9
140	306*	260*	6.8	600*	432*†	33	600*	373*	5.0
160	314*	301*	8.0	600*	359*	14			
180	600*	350*†	4.8	600*	557*†	5.3			
200	600*	†	5.8	600*	600*†	6.6			

*Solution terminated at 600 seconds for some or all instances.

†MILP solver ran out of memory for some or all instances, which are omitted from the average solution time.

Recent work – Benders for Scheduling

Segmented instances, wide time windows

Table 5: Average computation times in seconds for the segmented problem with wide time windows. The number of segments is 10% the number of jobs. Ten instances of each size are solved.

	Feasibility			Makespan			Tardiness		
Jobs	CP	MILP	Bndrs	CP	MILP	Bndrs	CP	MILP	Bndrs
60	0.05	12	1.9	0.2	16	5.8	0.2	8.0	2.3
80	0.28	22	2.5	180*	59	9.0	1.5	94	3.7
100	0.14	37	3.8	360*	403*	14	79*	594*	85*
120	0.13	61	5.0	540*	600*	25	600*	251*	183*
140	61*	175	7.0	600*	600*	107	600*	160*	4.3
160	540*	216*	4.8	600*	562*	157			
180	600*	375*†	4.5	600*	535*	10			
200	600*	†	5.5	600*	560*	6.9			

*Solution terminated at 600 seconds for some or all instances.
†MILP solver ran out of memory for some or all instances, which are omitted from the average solution time.

Recent work – Benders for Scheduling

Unsegmented instances

Table 6: Average computation times in seconds for the unsegmented problem. The number of segments is 10% the number of jobs. Ten instances of each size are solved,

	Feasibility			Makespan		
Jobs	CP	MILP	Bndrs	CP	MILP	Bndrs
60	0.10	11	2.8	0.2	24	5.1
80	0.14	21	3.7	0.7	376*	8.7
100	0.25	35	7.0	1.1	600*	21
120	0.43	57	23	0.4	600*	93
140	0.72	97	65	1.2	600*	115
160	420*	188	9.0	241*	549*	67
180	123*	307*	79	61*	600*	168
200	180*	410*	29	180*	587*	21

*Solution terminated at 600 seconds for some or all instances.

CP solves it quickly (< 1 sec) or blows up, in which case Benders solves it in 6 seconds (average).

So: try CP for 1 sec, then switch to Benders

Recent work – Cutting Planes from CP Model

Joint work with David Bergman.

Polyhedral analysis of overlapping all-different constraints (equivalent to graph coloring).

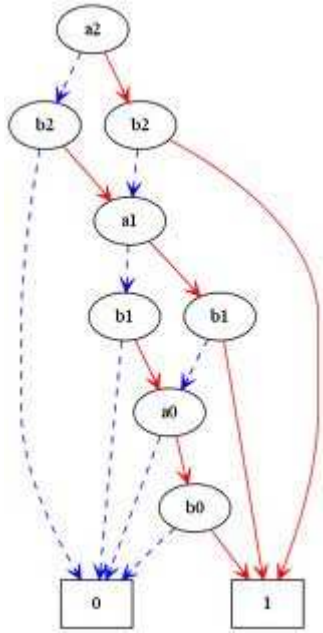
Used in many scheduling problems, sudoku puzzles, etc. etc.

Derive cutting planes from CP alldiff formulation and map them into 0-1 model.

Provides tighter bounds than all CPLEX cuts in a small fraction of the time (e.g., 1%).

Recent work – BDDs as Constraint Store

Joint work with Henrik Andersen, David Bergman, Andre Cire, Tarik Hadzic, Willem van Hoeve, Barry O'Sullivan, Peter Tiedemann



Replace variable domains in CP with relaxed **binary decision diagrams** (BDDs).

BDDs have long been used for circuit design, configuration, etc.

We use them to represent relaxation of feasible set.

Replace domain filtering with BDD-based propagation.

Reduces search tree for multiple alldiffs from 1 million nodes to 1 node, time speedup factor of 100. Speedups on other problems.

Now being incorporated into **Google CP solver**.

Recent work – BDDs for Relaxation Bounding

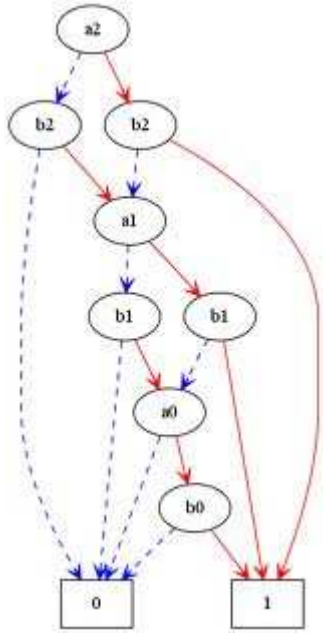
Joint work with David Bergman, Andre Cire, Willem van Hoeve

Replace LP relaxation with a relaxed **binary decision diagram** (BDD).

Shortest path in BDD provides a lower bound on optimal value.

For most instances of independent set problem, we get tighter bounds than full cutting plane technology in CPLEX.

Bound is normally obtained in very small fraction of the time.



Further Reading

J. N. Hooker, *Integrated Methods for Optimization*, 2nd ed., Springer, 2012.

M. Milano and P. van Hentenryck, eds., *Hybrid Optimization: The Ten Years of CPAIOR*, Springer, 2011.

J. N. Hooker, Operations research methods in constraint programming, in F. Rossi, P. van Beek and T. Walsh, eds., *Handbook of Constraint Programming*, Elsevier, 2006.