

# A Framework for Integrating Solution Methods

John Hooker

*Carnegie Mellon University*

ICS 2003

These slides will be available at

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

## Main Goals

- View different solution methods as special cases of a **single, general method**.
- In particular, unify **constraint programming, integer programming, and local search**.
- Do this by identify **common strategies** in these methods:
  - **Search** over problem restrictions
  - **Inference** of valid constraints
  - Solution of **relaxations** that guide the search

# Outline

- Knapsack illustration: Combining CP and IP.
- Common elements of CP, IP and local search.
- An integrated solver: putting it together
- Examples
  - Knapsack problem
  - Traveling salesman
  - Processing network design
  - Local search
  - Digression: relaxation and inference dualities
  - Benders decomposition
  - Machine scheduling – illustrates how Benders-based hybrid methods are a special case of present framework

## Illustration: Combining CP and IP

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

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

- constraint programming
- integer programming
- a hybrid approach

# Solve as a constraint programming problem

*Search:* Domain splitting

*Inference:* Domain reduction

*Relaxation:* Constraint store (set of current variable domains)

$$5x_1 + 8x_2 + 4x_3 \leq z$$

$$3x_1 + 5x_2 + 2x_3 \geq 30$$

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

$$x_j \in \{1, \dots, 4\}$$

Start with  $z = \infty$ .

Will decrease as feasible solutions are found.

Global constraint

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

## Domain reduction for inequalities

- Bounds propagation on  $5x_1 + 8x_2 + 4x_3 \leq z$   
 $3x_1 + 5x_2 + 2x_3 \geq 30$

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

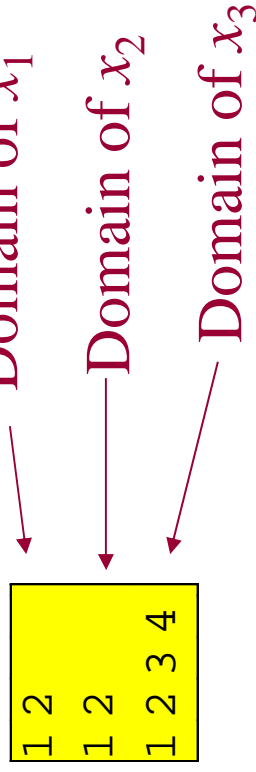
$$x_2 \geq \frac{30 - 3x_1 - 2x_3}{5} \geq \frac{30 - 12 - 8}{5} = 2$$

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

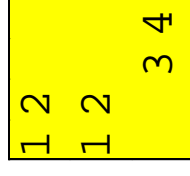
## Domain reduction for all-different (e.g., Régin)

- Maintain hyperarc consistency on all-different  $\{x_1, x_2, x_3\}$

Suppose for example:



Then one can reduce the domains:



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



Domain of  $x_1$   
 Domain of  $x_2$   
 Domain of  $x_3$

Domain of  $x_2$

$Z = \infty$

1	2	3	4
2	3	4	
1	2	3	4

$D_2 = \{2,3\}$

$Z = \infty$

3	4	
2	3	
2	3	4

$D_2 = \{2\}$

infeasible

$x = (4,3,2)$   
 value = 52

$Z = 52$

2	3
4	
1	2

$D_1 = \{2\}$

infeasible

$x = (3,4,1)$   
 value = 51

$D_1 = \{3\}$

$D_2 = \{4\}$

## Solve as an integer programming problem

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

*Inference:* Generate cutting planes (covering inequalities).

*Relaxation:* Continuous (LP) relaxation.

Rewrite problem using integer programming model:

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

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

## Continuous relaxation

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

Covering inequalities



## Branch and bound (Branch and relax)

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

At each node of the branching tree:

- If Optimal value of relaxation  $\geq$  Value of incumbent solution

There is no need to branch further.

- No feasible solution in that subtree can be better than the incumbent solution.
- Use SOS-1 branching.

$$y = \begin{bmatrix} 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 1/2 & 1/2 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$z = 49.5$

$y_{11} = 1$

$y_{12} = 1$

$y_{13} = 1$

$y_{14} = 1$

Infeas.

$$y = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1/2 & 0 & 1/2 & 0 \end{bmatrix}$$

$z = 50$

Infeas.

Infeas.

$$y = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0.2 & 0 & 0 & 0.8 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$z = 50.2$

Infeas.

Infeas.

Infeas.

Infeas.

$$y = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1/2 & 1/2 & 0 & 0 \end{bmatrix}$$

$z = 50$

Infeas.

$$y = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 2/15 & 0 & 0 & 13/15 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$z = 50.8$

Infeas.

$z = 54$

$$y = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0.1 & 0 & 0.9 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$z = 50.4$

Infeas.

$z = 52$

# Solve using a hybrid approach

## *Search:*

- Branch on fractional variables in solution of relaxation.
- Drop constraints with  $y_{ij}$ 's. This makes relaxation too large without much improvement in quality.
- If variables are all integral, branch by splitting domain.
- Use branch and bound.

## *Inference:*

- Use bounds propagation for all inequalities.
- Maintain hyperarc consistency for all-different constraints.

*Relaxation:*

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

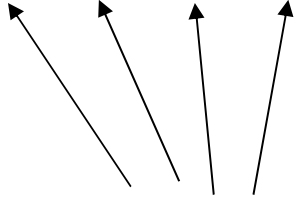


# Model for hybrid approach

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

Covering  
inequalities

Generate and  
propagate  
covering  
inequalities at  
each node of  
search tree



$Z = \infty$

1	2	3	4
2	3	4	
1	2	3	4

$x_2 = 3$

$x = (3.5, 3.5, 1)$   
value = 49.5

$x_2 = 4$

$Z = 52$

2	3	4
1	2	3

$x = (3.7, 3, 2)$   
value = 50.3

$x_1 = 3$

infeasible

$x_1 = 4$

$x = (4, 3, 2)$   
value = 52

$x_1 = 2$

$x = (2, 4, 3)$   
value = 54

$x_1 = 3$

$x = (3, 4, 1)$   
value = 51

$x = (2, 4, 2)$   
value = 50

## Advantages of Hybrid Approach

- CP brings:
  - Succinct and natural models
  - Ability to exploit “horizontal” structure (structure of subsets of constraints) by using *global* constraints
  - Constraint propagation technology
    - Particularly useful when constraints contain few variables (or objective is minmax)

## Advantages of Hybrid Approach

- IP brings:
  - Ability to exploit “vertical” structure (structure of special classes of problems)
  - Relaxation technology (cutting planes, Lagrangean relaxation, etc.)
  - Particularly useful when constraints contain many variables (or objective is min cost)
- Duality theory (Benders, Lagrangean dual, etc.)

# Common Elements of IP, CP and Local Search

- Search over problem restrictions
- CP: Search over partial solutions
  - Branch on variable domains
  - Domains can be finite or continuous intervals
- IP: Search over restrictions
  - Branch on fractional variables
- Local search: Search over complete solutions
  - Move to a neighboring solution

# Common Elements of IP, CP and Local Search

- Inference of new constraints
- CP: Domain reduction / consistency maintenance
  - In-domain constraints for finite domains ( $x_j \in D_j$ )
    - Ideal: *hyperarc consistency* (remove all elements of domain that are not part of some solution)
  - Interval arithmetic
    - Ideal: *bounds consistency* (shrink interval domains as much as possible)
- IP: Cutting planes
  - Covering inequalities, etc., etc.
    - Ideal: facet-defining cuts
- Local search: research topic

# Common Elements of IP, CP and Local Search

- **Relaxation**
- **CP: constraint store**
  - Primarily in-domain constraints
  - Constraint store is not “solved” but propagates constraints and guides branching
- **IP: continuous relaxation**
  - Inequalities in continuous variables
  - Solution of relaxation provides bounds and guides branching
- **Local Search: research topic**

## An Integrated Solver

- A *recursion* specifies **search** over problem restrictions.
- An *inference* engine **infers** valid constraints for each problem restriction.
- **Relaxations** and *special-purpose solvers* provide search guidance and perhaps bounds to prune the search.



## Exploiting Structure

- The **search** process can be guided to suit the problem by setting parameters in canned recursive procedures.
- Specialized **domain reduction** and **cutting plane** procedures can be designed for global constraints (which represent specially-structured subsets of constraints)
  - This makes it convenient to use **existing cutting-plane technology**
- Specialized **relaxations** can be designed for global constraints and assembled into one or more relaxations of the entire problem.

# Putting It Together

- Model consists of
  - *declaration window* (variables, initial domains)
  - *relaxation windows* (initialize relaxations & solvers)
  - *constraint windows* (each with its own syntax)
  - *objective function* (optional)
  - *search window* (invokes propagation, branching, relaxation, etc.)
- Basic algorithm searches over problem restrictions, drawing inferences and solving relaxations for each.

## Putting It Together

- Relaxations may include:
  - Constraint store (with domains)
  - Linear programming relaxation, etc.
- The relaxations link the windows.
  - Propagation (e.g., through constraint store).
  - Search decisions (e.g., nonintegral solutions of linear relaxation).

# Putting It Together

- A generic algorithm:
  - **Process constraints.**
    - Infer new constraints, reduce domains & propagate, generate relaxations.
  - **Solve relaxations.**
    - Check for empty domains, solve LP, etc.
  - **Continue search (recursively).**
    - Create new problem restrictions if desired (e.g, new tree branches).
    - Select problem restriction to explore next (e.g., backtrack or move deeper in the tree).

## Example

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

## Declaration Window

$x_j \in \{1, 2, 3, 4\}, j = 1, \dots, 4$

Variables and initial  
domains

## Objective Function Window

$$\min 5x_1 + 8x_2 + 4x_3$$

## Relaxation Window

*Type:* **Constraint store**, consisting of variable domains.

*Objective function:* None.

*Solver:* None.



## Relaxation Window

*Type:* **Linear programming.**

*Objective function:* Same as original problem.

*Solver:* LP solver.

## Constraint Window

*Type:* **Linear (in)equalities.**

$$3x_1 + 5x_2 + 2x_3 \geq 30$$

*Inference:* Bounds consistency maintenance.

*Inference:* Covering inequalities using all-different constraint.

*Relaxation:* Add reduced bounds to constraint store.

*Relaxation:* Add original inequality and covering inequalities to LP relaxation.

## Constraint Window

*Type:* **All-different**

all-different( $x_1, x_2, x_3, x_4$ )

*Inference:* Régin's hyperarc consistency maintenance.

*Relaxation:* None. (A relaxation exists but is not helpful in this case.)

## Search Window

**Procedure BandBsearch( $P, R, S, \text{CustomBranch}$ )**  
(canned branch & bound search using CustomBranch  
as branching rule)

## User-Defined Window

### **Procedure CustomBranch( $P, R, S, i$ )**

[Take the  $i$ -th branch for problem restriction  $P$ , whose relaxation  $R$  has solution  $S$ ]

If there is a variable with nonintegral value in LP relaxation then

    Perform **BranchOnFraction**( $P, R, S, i$ ) [*standard B&B branching*]

    Else perform **FirstFail**( $P, R, S, i$ ) [*Standard domain splitting*]

## Example: Traveling Salesman

$$\begin{aligned} \min \quad & \sum_j^c y_j y_{j+1} \\ \text{s.t.} \quad & \text{all - different } \{y_1, \dots, y_n\} \end{aligned}$$

$$y_j \in \{1, \dots, n\}$$

$j$ -th city in tour

or

enforces  
Hamiltonian  
cycle

$$\begin{aligned} \min \quad & \sum_j^c y_j \\ \text{s.t.} \quad & \text{cycle} \{y_1, \dots, y_n\} \end{aligned}$$

$$y_j \in \{1, \dots, n\}$$

city following  $j$   
in tour

## *Element* global constraint

To implement a variably indexed constant  $a_y$

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

Domain reduction is trivial.

Convex hull relaxation of element constraint is simply

$$\min_{j \in D_y} \{a_j\} \leq z \leq \max_{j \in D_y} \{a_j\}$$

Current domain of  $y$



## Extension of *element*

To implement variably indexed variable  $y_x$

Replace  $y_x$  with  $z$  and add constraint  $\text{element}(y, (x_1, \dots, x_n), z)$   
which posts constraint  $\bigvee_{j \in D_y} (z = x_j)$

Domain reduction is fairly straightforward.



Relaxation is based on relaxation of disjunctive constraint:

If  $0 \leq x_j \leq m_0$  for each  $j$ , there is a simple convex hull relaxation (*JNH*):

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

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

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

## Declaration Window for TSP

$y_j \in D_j = \{2, \dots, n\}$ ,  $j = 2, \dots, n$        $j$ th city in tour

$y_1 \in D_1 = \{1\}$

$z_j \in R$  for  $j = 1, \dots, n$       cost of  $j$ th link in tour

# Objective Function Window

$$\min \sum_{j=1}^n z_j$$

## Relaxation Window

*Type:* **Constraint store**, consisting of domains of  $y_1, \dots, y_n$

*Objective function:* None.

*Solver:* None.

## Relaxation Window

*Type: Linear programming.*

*Objective function:  $\min \sum_{jk} c_{jk} x_{jk}$*

*Solver: LP solver.*

## Constraint Window

**Type: Element.**

element( $y_j, (c_{j1}, \dots, c_{jn}), z_j$ ) for  $j = 1, \dots, n$

*Inference:* Hyperarc consistency maintenance.

*Relaxation:* Add reduced bounds to constraint store.

*Relaxation:* Add disjunctive relaxation to LP.

## Constraint Window

*Type:* **cycle**

$\text{cycle}(y_1, \dots, y_n)$

*Inference:* Domain reduction (research topic).

*Relaxation:* Add reduced domains to constraint store.

*Relaxation:* Standard IP relaxation and cutting planes for TSP. Also fix  $x_{jk} = 0$  if  $k \notin D_j$  and  $x_{jk} = 1$  if  $D_j = \{k\}$ .

## Search Window

**Procedure BandBsearch( $P, R, S, \text{TSPBranch}$ )**  
(canned B&B search using TSPBranch as branching rule)



## User-Defined Window

### Procedure TSPBranch( $P, R, S, i$ )

[*Take the  $i$ -th branch*]

If there is a variable  $x_{jk}$  with a nonintegral value in LP relaxation then

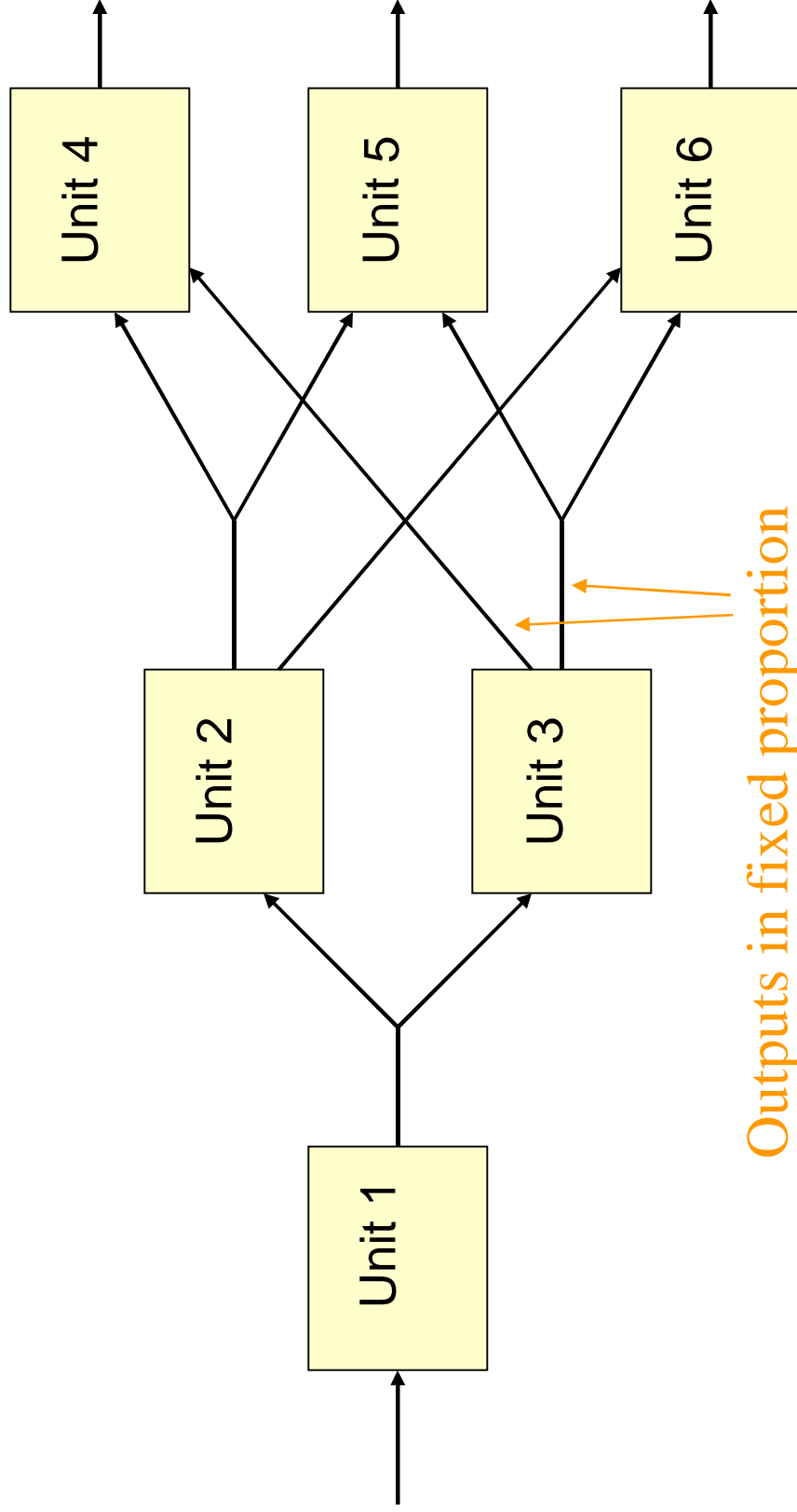
    If  $i = 1$  then create  $P'$  from  $P$  by letting  $D_j = \{k\}$  and return  $P'$ .

    If  $i = 2$  then create  $P'$  from  $P$  by letting  $D_j = D_j \setminus \{k\}$  and return  $P'$ .

## Example: Processing Network Design

- Find optimal design of processing network.
- A “superstructure” (largest possible network) is given, but not all processing units are needed.
- Internal units generate negative profit.
- Output units generate positive profit.
- Installation of units incurs fixed costs.
- Objective is to maximize net profit.

# Sample Processing Superstructure



## Declaration Window

$u_i \in [0, c_i]$

flow through unit  $i$

$x_{ij} \in [0, c_{ij}]$

flow on arc  $(i, j)$

$z_i \in [0, \infty]$

fixed cost of unit  $i$

$y_i \in D_i = \{\text{true}, \text{false}\}$

presence or absence of unit  $i$

# Objective Function Window

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

Net revenue generated by unit  $i$  per unit flow

## Relaxation Window

*Type:* **Constraint store**, consisting of variable domains.

*Objective function:* None.

*Solver:* None.

## Relaxation Window

**Type: Linear programming.**

**Objective function:** Same as original problem.

**Solver:** LP solver.

## Constraint Window

*Type:* **Linear (in)equalities.**

$Ax + Bu = b$  (flow balance equations)

*Inference:* Bounds consistency maintenance.

*Relaxation:* Add reduced bounds to constraint store.

*Relaxation:* Add equations to LP relaxation.



## Constraint Window

*Type:* **Disjunction of linear inequalities.**

$$\left( \begin{array}{l} y_i \\ z_i \geq d_i \end{array} \right) \vee \left( \begin{array}{l} \neg y_i \\ u_i \leq 0 \end{array} \right) \quad \text{for each } i$$

*Inference:* None.

*Relaxation:* Add convex hull relaxation to LP.

## Constraint Window

**Type: Propositional logic.**

Don't-be-stupid constraints:

$$y_1 \rightarrow (y_2 \vee y_3) \quad y_3 \rightarrow y_4$$

$$y_2 \rightarrow y_1 \quad y_3 \rightarrow (y_5 \vee y_6)$$

$$y_2 \rightarrow (y_4 \vee y_5) \quad y_4 \rightarrow (y_2 \vee y_3)$$

$$y_2 \rightarrow y_6 \quad y_5 \rightarrow (y_2 \vee y_3)$$

$$y_3 \rightarrow y_1 \quad y_6 \rightarrow (y_2 \vee y_3)$$

**Inference:** Resolution (add resolvents to constraint set).

**Relaxation:** Add reduced domains of  $y_i$ 's to constraint store.

**Relaxation (optional):** Add 0-1 inequalities representing propositions to LP.

## Search Window

**Procedure BandBsearch( $P, R, S, \text{NetBranch}$ )** (canned  
branch & bound search using **NetBranch** as  
branching rule)

## User-Defined Window

### **Procedure NetBranch( $P, R, S, i$ )**

Let  $i$  be a unit for which  $u_i > 0$  and  $z_i < d_i$ .

If  $i = 1$  then create  $P'$  from  $P$  by letting  $D_i = \{T\}$   
and return  $P'$ .

If  $i = 2$  then create  $P'$  from  $P$  by letting  $D_i = \{F\}$   
and return  $P'$ .

# Example: Local Search

## Allocation Problem (Williams)

Each retailer  $j$  is served by division 1 ( $x_j = 1$ ) or 2 ( $x_j = 0$ ).

Division 1 delivers  $a_{ij}$  units of product  $i$  to retailer  $j$ .

Assign divisions so as to match division 1 quotas  $b_i$  as closely as possible.

$$\begin{aligned} \min \quad & \sum_i |s_i| \\ \text{s.t.} \quad & \sum_j a_{ij} x_j + s_i = b_i, \quad \text{all } i \\ & x_j \in \{0,1\}, \quad s_i \in R \end{aligned}$$

Known to be very hard for IP (Cornuejols & Dawande 1999)

## Declaration Window

$x_j \in D_j = \{0\}$

1 when division 1 is assigned  
to retailer  $j$

$s_i \in R$

error in meeting goal for  
product  $i$

# Objective Function Window

$$\min f(x) = \sum_i |s_i|$$

## Relaxation Window

*Type:* **Linear.**

*Objective function:* Same as original problem.

*Solver:* Direct computation (with efficient updating).



## Constraint Window

*Type:* **Linear equations.**

$$s_i = b_i - \sum_j a_{ij} \cdot x_j$$

*Inference:* None.

*Relaxation:* Compute  $s_i$  as above, with each  $x_j$  fixed to the single value in  $D_j$ .

## Search Window

### **Procedure AnnealingSearch( $P, R, S$ )**

Return if search has run long enough.

Let  $\text{random}(p)$  be a random variable that has value 1 with probability  $p$ .

To flip a domain  $D_j = \{t\}$  is to change it to  $\{1 - t\}$ .

“Solve” relaxation to get objective value  $v$ .

Do forever:

    Randomly select  $j$  and change  $P$  by flipping  $D_j$ .

    Solve relaxation to get new objective value  $v'$ .

    If  $v' < v$  or  $\text{random}(p) = 1$  then

        Perform **AnnealingSearch**( $P, R, S$ ) and return.

    Restore  $P$  by flipping  $D_j$ .

## Example: Benders Decomposition

- Benders decomposition (and generalizations of it) fit into the framework.
- The master problem becomes a relaxation.
- The search is over subproblems, which are restrictions of the original problems.
- Will apply a generalized Benders to a machine scheduling problem.

The problem:

$$\begin{array}{ll} \min & cx + dy \\ \text{s.t.} & Ax + By \geq a \\ & x \in R^n, y \in Z^m \end{array}$$

The subproblem:

$$\begin{array}{ll} \min & cx + d\bar{y} \\ \text{s.t.} & Ax \geq a - B\bar{y} \quad (u) \end{array}$$

$\bar{y}$  = solution of previous master problem

The master problem:

$$\begin{array}{ll} \min & z \\ \text{s.t.} & z \geq \bar{u}(a - B\bar{y}) + dy \\ & + \text{other Benders cuts} \end{array}$$

Assume subproblem is feasible and bounded.

# Declaration Window

$x_i \in R$

subproblem variables

$y_j \in Z$

master problem variables

# Objective Function Window

$$\min \quad cx + dy$$

## Relaxation Window

*Type:* **MILP** (master problem).

*Objective function:* minimize  $z$

*Solver:* IP solver

## Constraint Window

*Type: Linear inequalities.*

$$Ax + By \geq a$$

*Inference: Generation of Benders cuts from subproblem dual.*

*Relaxation: Add Benders cuts to IP relaxation (master problem).*



## Search Window

### **Procedure BendersSearch( $P, R, S$ )**

Generate Benders cut from  $P$  (subproblem).

Find optimal value  $v$  of relaxation (master problem).

If  $v$  is equal to optimal value of  $P$ , stop.

Define next subproblem.

Obtain  $P'$  from  $P$  by setting each  $D_j$  to  $\{\bar{y}_j\}$

Perform **BendersSearch( $P', R, S$ )**.

## Benders-based Combination of CP and IP

- One promising scheme for combining CP and IP is based on generalized Benders decomposition.
- IP solves master problem, CP solves subproblem.
- Subproblem “dual” is inference dual, which generalizes LP dual.
- This scheme is a special case of the framework presented here.
- Will apply it to a machine scheduling problem.

## Digression: Dualities

Two related dualities tend to occur in optimization.

- **Relaxation duality** (relaxations are parameterized).

- Dual problem is to search parameter space for strongest relaxation.
- Search parameter space for tightest relaxation.
- Examples: LP, Lagrangean and surrogate duality.

- **Inference duality**

- Dual problem is to infer valid constraints.
- Examples: LP dual, dual used in scheduling problem below.

## Digression: Relaxation Duality

The problem:  $\min_{x \in S} \{f(x)\}$

Parameterized relaxation:

$$\theta(\lambda) = \min_{x \in S(\lambda)} \{f(x, \lambda)\}$$

Dual problem:

$$\max_{\lambda \in \Lambda} \{\theta(\lambda)\}$$

Example: Lagrangean (& LP) duality  $\min_{x \in S} \{f(x) \mid g(x) \leq 0\}$

Parameterized relaxation:

$$\theta(\lambda) = \min_{x \in S} \{f(x) + \lambda^T g(x)\}$$

Dual problem:

$$\max_{\lambda \geq 0} \{\theta(\lambda)\}$$

# Digression: Inference Duality

The problem:  $\min_{x \in S} \{f(x)\}$

Dual problem:

$\max_{\text{proofs}} \{z \mid x \in S \Rightarrow f(x) \geq z\}$

$Ax \geq b \Rightarrow cx \geq z$  when  
 $uAx \geq ub$  dominates  $cx \geq z$ ;  
that is, when  $uA \leq c$  and  $ub \geq z$

Example: LP duality

$\min\{cx \mid Ax \geq b\}$

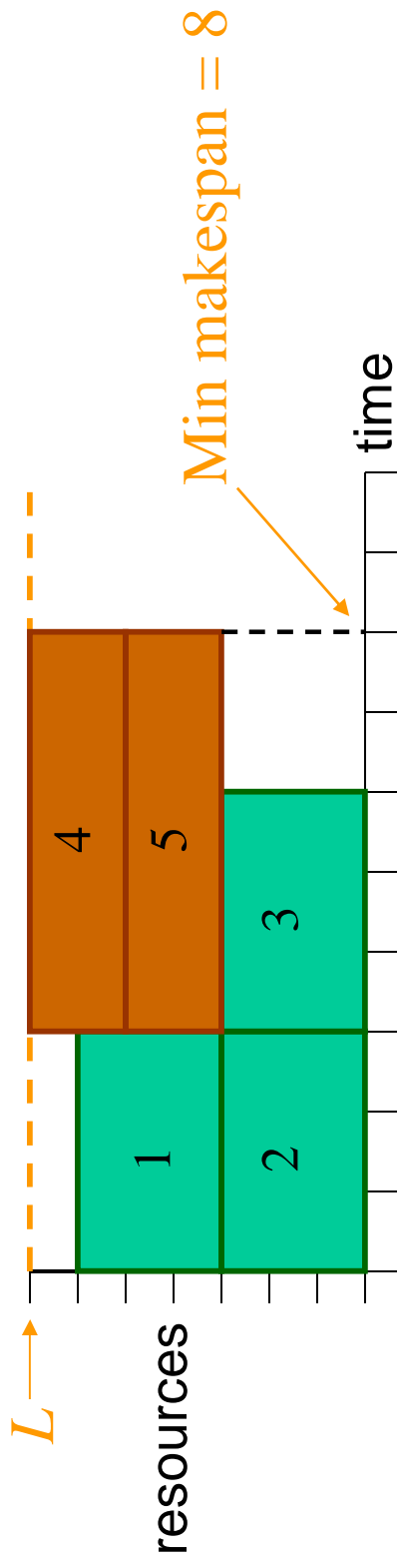
Dual problem:

$\max_{\text{proofs}} \{z \mid Ax \geq b \Rightarrow cx \geq z\} = \max_{u \geq 0} \{ub \mid uA \leq c\}$



# Cumulative Global Constraint

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



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

$L \rightarrow$   
 Resources used  
 Durations  
 Job start times

# Machine scheduling

Schedule jobs on parallel machines.

- Machines run at different speeds and incur different costs per job.
- Each job has a release date and a due date.
- Master problem assigns jobs to machines.
- Subproblem schedules jobs on assigned machines.

## Domain reduction

Highly developed; based on edge finding.

### Relaxation (JNH & Yan, 2001)

If some subset of jobs  $\{j_1, \dots, j_k\}$  are identical (same release time  $a_0$ , duration  $d_0$ , and resource consumption rate  $r_0$ ), then

$$t_{j_1} + \dots + t_{j_k} \geq (P+1)a_0 + \frac{1}{2}P[2k - (P+1)Q]d_0$$

is a valid cut and is facet-defining if there are no deadlines, where

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



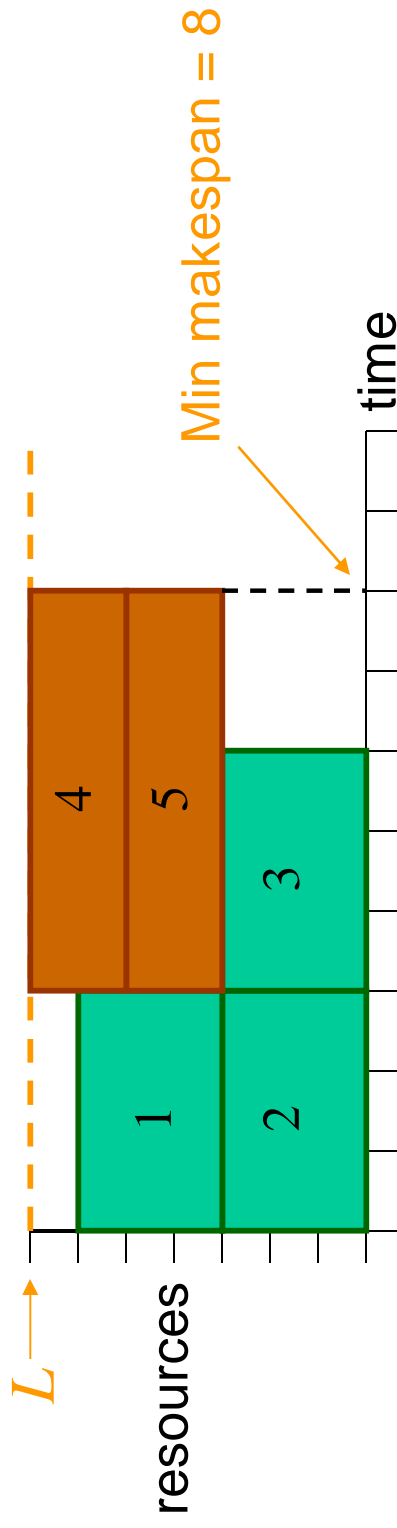
The following cut is valid for any subset of jobs  $\{j_1, \dots, j_k\}$

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

Where the jobs are ordered by nondecreasing  $r_j d_j$ .

Analogous cuts can be based on deadlines.

## Example



min  $z$

s.t.  $z \geq t_1 + 3, t_2 + 3, t_3 + 3, t_4 + 5, t_5 + 5$

$t_1 + t_2 + t_3 \geq 3$

$t_1 + t_2 + t_3 + t_4 \geq 3 \frac{5}{14}$

$t_2 + t_3 + t_4 + t_5 \geq 2 \frac{4}{7}$

$t_1 + t_2 + t_3 + t_4 + t_5 \geq 6 \frac{6}{7}$

$t_j \geq 0$

Facet defining

Resulting bound:

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

A model for the machine scheduling problem:

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

Cost of assigning machine  $x_j$  to job  $j$

Release date for job  $j$

Job duration

Deadline

Start time for job  $j$

Machine assigned to job  $j$

Start times of jobs assigned to machine  $i$

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

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

Feasibility of each problem is checked by constraint programming. One or more infeasible problems results in an optimal value  $\infty$ . Otherwise the value is zero.

Suppose there is no feasible schedule for machine  $i$ . Then some subset  $J_i(\bar{x})$  of jobs cannot be assigned to machine  $i$ . We have a Benders cut

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

This yields the master problem,

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

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

$$\begin{aligned}
& \min \quad \sum_{ij} C_{ij} y_{ij} \\
& \text{s.t.} \quad t_j \geq R_j, \quad \text{all } j \\
& \quad \quad t_j + \sum_i D_{ij} y_{ij} \leq S_j, \quad \text{all } j \\
& \quad \quad \sum_i y_{ij} \geq 1, \quad \text{all } j \\
& \quad \quad \sum_j (1 - y_{ij}) \geq 1, \quad \text{all } i, \quad k = 1, \dots, K
\end{aligned}$$

Valid

constraint

added to  $\sum_{j^k=i} D_{ij} y_{ij} \leq \max_j \{S_j\} - \min_j \{R_j\}$ , all  $i$

improve

performance

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