A Modeling System to Combine Optimization and Constraint Programming

John Hooker
Carnegie Mellon University

INFORMS, November 2000

Based on joint work with…

- Ignacio Grossmann
- Hak-Jin Kim
- Maria Auxilio Osorio
- Greger Ottosson
- Erlendur Thorsteinsson

Goal

Design a modeling language whose syntax indicates how optimization and constraint programming can combine to solve the problem.

Why Combine Optimization and Constraint Programming?

- Some constraints in a given problem propagate well and others relax well.
    - Constraint programming uses *constraint propagation* to reduce domains (sets of possible values) of variables and so reduce branching.
    - Optimization uses easily-solved *relaxations* to obtain bounds on the optimal value and so reduce branching.
- Constraint programming uses *global constraints* to inform the solver about special structure in the model.

General Form of Model

min	$f(x)$	objective function
s.t.	$q_i(y) \rightarrow h_i(x), \quad i \in I_3$	conditional constraints
	$x \in X$	solution variables
	$y_j \in D_j, \quad \text{all } j$	search variables

Checkable constraint (belongs to NP)

Set of soluble constraints (belong to NP $\cap$ co-NP)

Usually continuous

Usually discrete

Additional Types of Constraints

$p_i(y), \quad i \in I_1$	checkable constraints
$g_i(x), \quad i \in I_2$	soluble constraints
$d_i(x, y), \quad i \in I_4$	global constraints

Degenerate cases of a conditional constraint

Definable in terms of a set of conditional constraints

The Basic Idea

- Branch on search variables.
- At each node of the search tree:
  - Apply constraint propagation to checkable and global constraints to reduce domains.
  - Create a relaxation.
    - If antecedent of a conditional is true, “fire” the conditional by adding consequent to relaxation.
    - Add relaxations of global constraints to relaxation.
  - Use optimal value of relaxation to prune tree if possible.
  - Backtrack, or try to find feasible values of search variables consistent with solution variables

Relaxation at a Given Node of the Search Tree

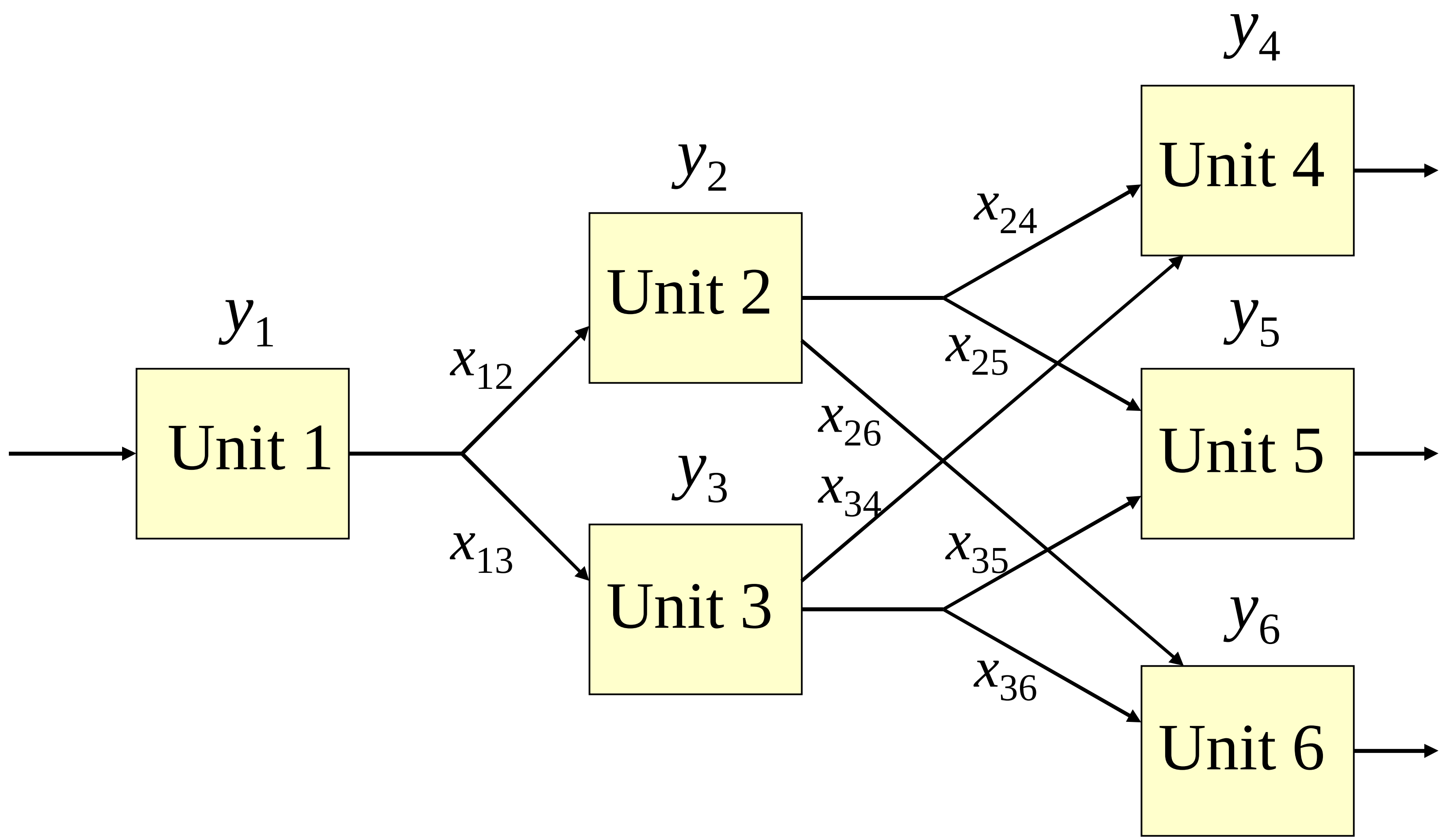
Soluble constraints

Consequents of "fired" conditional constraints

$$
\begin{array}{ll}
\min & f(x) \\
\text{s.t.} & g_i(x), \quad i \in I_2 \\
 & h_i(x), \quad \text{all } i \in I_3 \text{ for which } q_i(y) \text{ is true} \\
 & \text{relaxation of } d_i(x,y), \quad i \in I_4 \\
 & x \in X
\end{array}
$$

Specialized relaxations of global constraints

Processing Network Design



Processing Network Design

- The model uses search variables y_i to indicate the presence or absence of a unit.
- It uses conditional constraints to require that the fixed cost be incurred or the unit shut down.

$$0.6u_2 = x_{24} + x_{25}$$
$$0.4u_2 = x_{26}$$
$$0.7u_3 = x_{34}$$
$$0.3u_3 = x_{35} + x_{36}$$

$$
\begin{array}{lll}
\max & \sum_i r_i u_i^{1/2} - \sum_i z_i & \\
\text{s.t.} & u = Ax & \text{flow thru units} \\
& bu = Bx & \text{flow balance} \\
& (y_i = \text{true}) \rightarrow (z_i = d_i),\ \text{all } i & \text{unit is open} \\
& (y_i = \text{false}) \rightarrow (u_i = 0),\ \text{all } i & \text{unit is closed} \\
& u \le c & \text{unit capacities} \\
& u, x \ge 0 &
\end{array}
$$

Processing Network Design

- Add don’t-be-stupid constraints to ensure that a unit is not opened unless downstream units are opened.

$$
\begin{array}{lll}
\max & \sum_i r_i u_i^{1/2} - \sum_i z_i & \\
\text{s.t.} & u = Ax & \text{flow thru units} \\
 & bu = Bx & \text{flow balance} \\
 & (y_i = \text{true}) \rightarrow (z_i = d_i), \text{ all } i & \text{unit is open} \\
 & (y_i = \text{false}) \rightarrow (u_i = 0), \text{ all } i & \text{unit is closed} \\
 & u \le c & \text{unit capacities} \\
 & u, x \ge 0 & \\
 & \left\{ \begin{array}{ll}
 y_1 \rightarrow (y_2 \vee y_3) & y_3 \rightarrow y_4 \\
 y_2 \rightarrow y_1 & y_3 \rightarrow (y_5 \vee y_6) \\
 y_2 \rightarrow (y_4 \vee y_5) & y_4 \rightarrow (y_2 \vee y_3) \\
 y_2 \rightarrow y_6 & y_5 \rightarrow (y_2 \vee y_3) \\
 y_3 \rightarrow y_1 & y_6 \rightarrow (y_2 \vee y_3)
 \end{array} \right\} & \text{don't - be - stupid}
\end{array}
$$

Processing Network Design

- Use an *inequality-or* global constraint to obtain good relaxation of disjunctive constraints.
- Use *cnf* global constraint to invoke resolution algorithm for don’t-be-stupid constraints.

Part of relaxation

Include relaxation of global constraint

$$\max \sum_i r_i u_i^{1/2} - \sum_i z_i$$

s.t.

$$u = Ax \qquad \text{flow thru units}$$

$$bu = Bx \qquad \text{flow balance}$$

$$\text{inequality-or}\left(\begin{bmatrix} y_i \\ \neg y_i \end{bmatrix}, \begin{bmatrix} z_i \geq d_i \\ u_i = 0 \end{bmatrix} \right) \qquad \text{global constraint}$$

$$u \leq c \qquad \text{unit capacities}$$

$$u, x \geq 0$$

$$\text{cnf}\begin{pmatrix} y_1 \rightarrow (y_2 \vee y_3) & y_3 \rightarrow y_4 \\ y_2 \rightarrow y_1 & y_3 \rightarrow (y_5 \vee y_6) \\ y_2 \rightarrow (y_4 \vee y_5) & y_4 \rightarrow (y_2 \vee y_3) \\ y_2 \rightarrow y_6 & y_5 \rightarrow (y_2 \vee y_3) \\ y_3 \rightarrow y_1 & y_6 \rightarrow (y_2 \vee y_3) \end{pmatrix} \qquad \text{global constraint}$$

Process symbolically

Knapsack Problem with All-different

Original problem

$$
\begin{aligned}
\min\quad & 5y_1 + 8y_2 + 4y_3 \\
\text{s.t.}\quad & 3y_1 + 5y_2 + 2y_3 \geq 30 \\
& \text{all-different}(y_1, y_2, y_3) \\
& y_j \in \{1,2,3,4\}, \quad \text{all } j
\end{aligned}
$$

As modeled here, solved by branching and domain reduction only.

Knapsack Problem with All-different

The *continuous* predicate adds a continuous relaxation and any desired cutting planes.

$$\begin{aligned} \min\ & \text{continuous}(5y_1 + 8y_2 + 4y_3) \\ \text{s.t.}\ & \text{continuous}(3y_1 + 5y_2 + 2y_3 \geq 30) \\ & \text{all-different}(y_1, y_2, y_3) \\ & y_j \in \{1,2,3,4\}, \quad \text{all } j \end{aligned}$$

Replace objective function with $5x_1 + 8x_2 + 4x_3$

Add $3x_1 + 5x_2 + 2x_3 \geq 30$ and link(y_j,x_j) and possibly knapsack cuts

Knapsack Problem with All-different

The *cut* predicate generates cuts in the search variables so that domain reduction is applied to cuts. *Continuous* adds continuous relaxation of problem and cuts.

$$
\begin{array}{ll}
\min & z \\
\text{s.t.} & \text{continuous}\left(\text{cut}\begin{pmatrix} z \geq 5y_1 + 8y_2 + 4y_3 \\ 3y_1 + 5y_2 + 2y_3 \geq 30 \end{pmatrix} \right) \\
& \text{all-different}(y_1, y_2, y_3) \\
& y_j \in \{1,2,3,4\}, \quad \text{all } j
\end{array}
$$

Cumulative Global Constraint

Ensures that total resources consumed by jobs at any one time do not exceed C.

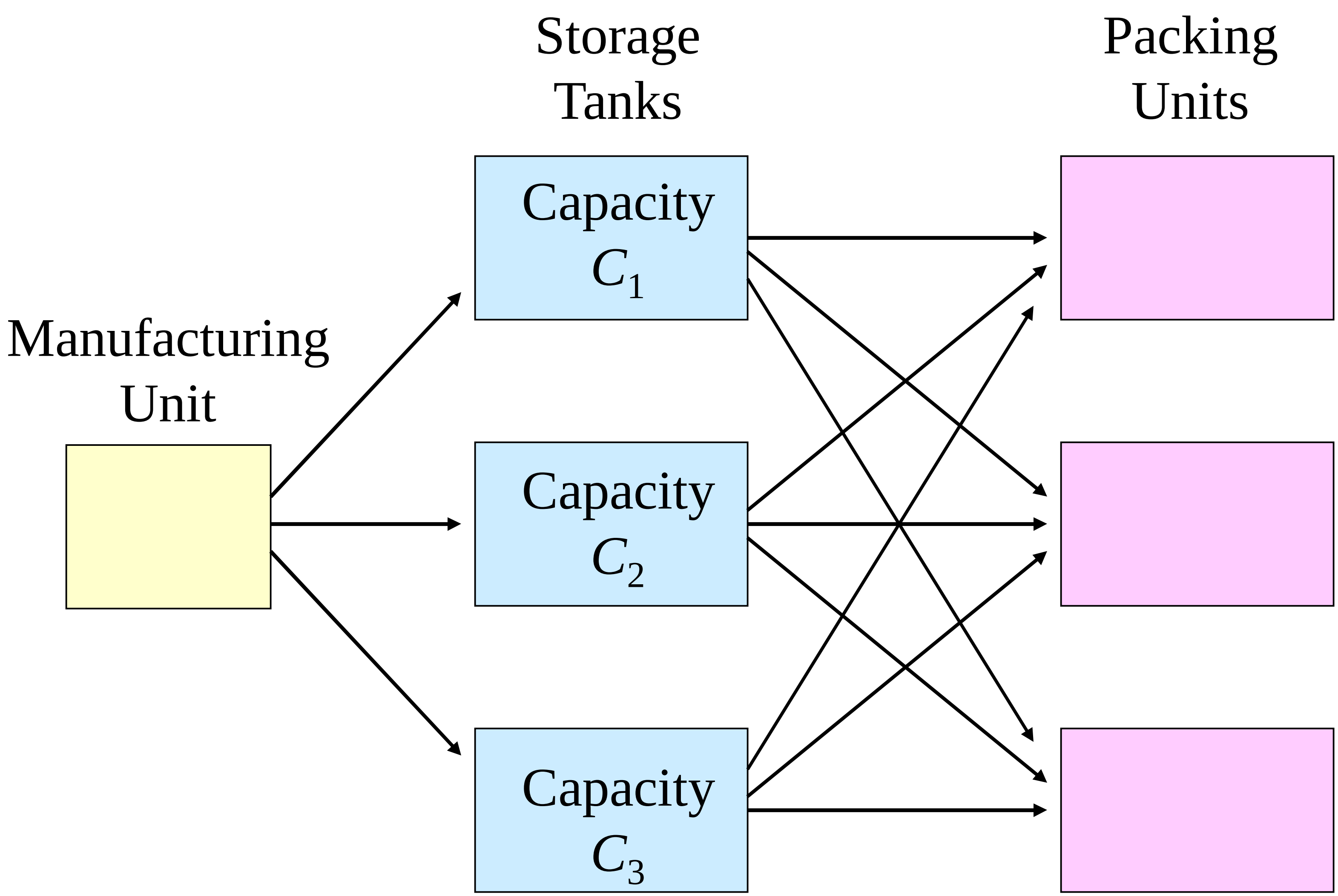
$$\text{cumulative}\big((t_1,\ldots,t_n),(d_1,\ldots,d_n),(r_1,\ldots,r_n),C\big)$$

Job start times

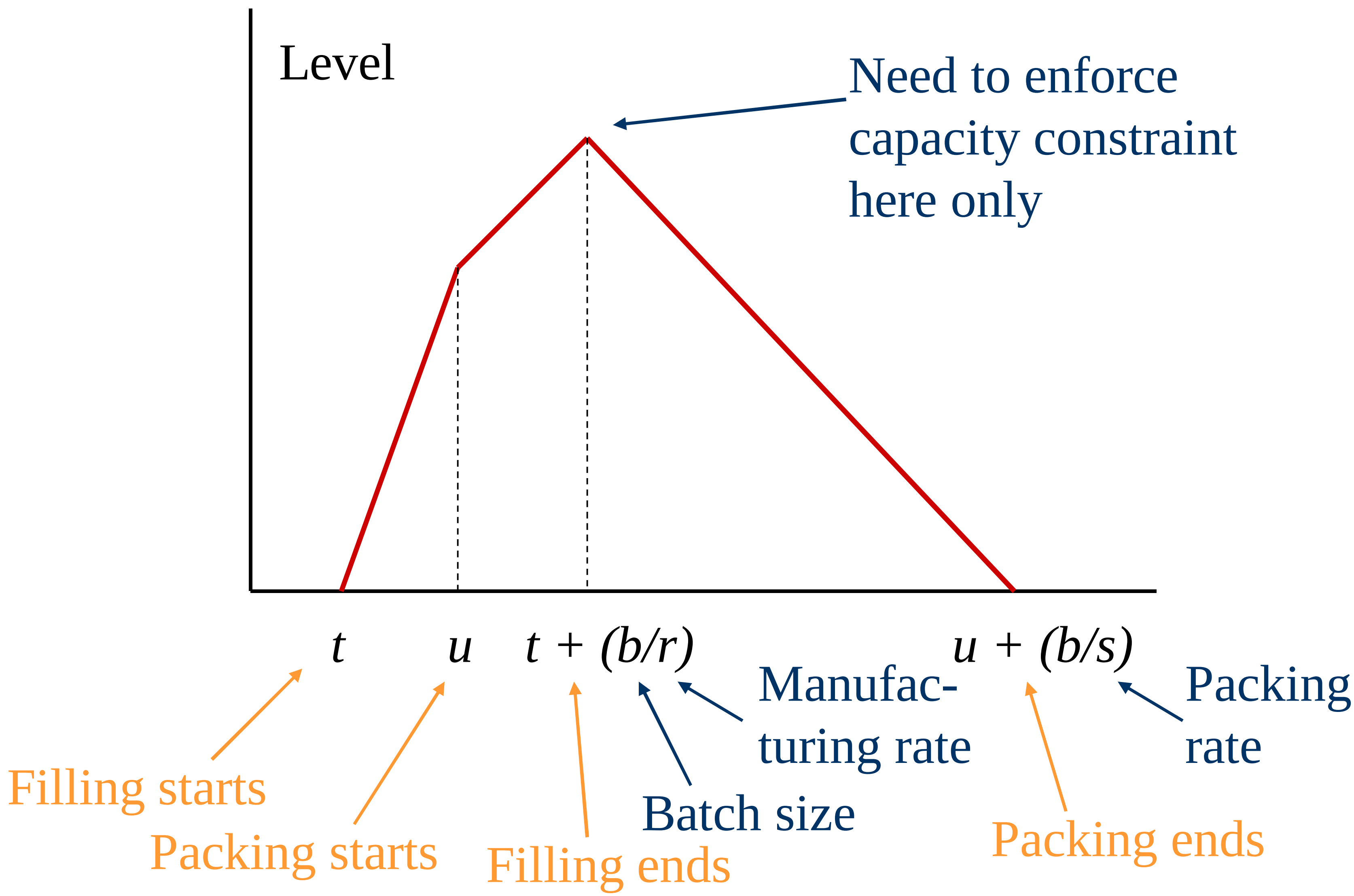
Job durations

Job resource requirements

Production Scheduling



Filling of StorageTank



$$
\begin{aligned}
\min \quad & T && \leftarrow \text{Makespan} \\
\text{s.t.} \quad & T \ge u_j + \frac{b_j}{s_j}, \quad \text{all } j \\
& t_j \ge R_j, \quad \text{all } j && \leftarrow \text{Job release time} \\
& \text{cumulative}\left(t, v, (1,\ldots,1), m\right) && \leftarrow m \text{ storage tanks} \\
& v_i = u_i + \frac{b_i}{s_i} - t_i, \quad \text{all } i && \leftarrow \text{Job duration} \\
& b_i\left(1 - \frac{s_i}{r_i}\right) + s_i u_i \le C_i, \quad \text{all } i && \leftarrow \text{Tank capacity} \\
& \text{cumulative}\left(u, \left(\frac{b_1}{s_1}, \ldots, \frac{b_n}{s_n}\right), e, p\right) && \leftarrow p \text{ packing units} \\
& u_j \ge t_j \ge 0
\end{aligned}
$$

$$
\begin{aligned}
\text{min} \quad & T \\
\text{s.t.} \quad & T \ge u_j + \frac{b_j}{s_j}, \quad \text{all } j \\
& t_j \ge R_j, \quad \text{all } j \\
& \text{cumulative}\big(t, v, (1,\ldots,1), m\big) \\
& v_i = u_i + \frac{b_i}{s_i} - t_i, \quad \text{all } i \\
& b_i\left(1 - \frac{s_i}{r_i}\right) + s_i u_i \le C_i, \quad \text{all } i \\
& \text{cumulative}\left(u, \left(\frac{b_1}{s_1}, \ldots, \frac{b_n}{s_n}\right), e, p\right) \\
& u_j \ge t_j \ge 0
\end{aligned}
$$

Part of relaxation

Apply domain reduction (relaxation yet to be developed)