Topics:

- Rethinking of integer variables
- Links between optimization and constraint programming.
- Making links in a declarative modeling language.

Much of the following is joint work with...

Maria A. Osorio
Hak-Jin Kim
Gregor Ottosson.

Modeling without integer variables

TSP: $\min \sum_j c_{y_j y_{j+1}}$ ← variable subscripts

s.t. all-different $\{y_1, \ldots, y_n\}$ ← 1st city in tour

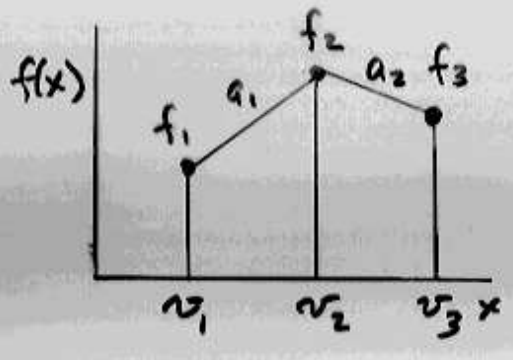
$y_j \in \{1, \ldots, n\}$.

QAP: $\min \sum_{j,k} c_{jk} d_{y_j y_k}$ ← cost ← distance

s.t. all-different $\{y_1, \ldots, y_n\}$ ← location assigned facility 1

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

Piecewise linear function:



$$f(x) = f_y + a_y(x - v_y)$$

$$v_y \le x \le v_{y+1}$$

$$y \in \{1, 2\}.$$

1-machine scheduling:

$$\min \sum_j z_j$$ ← total tardiness

$$\text{s.t.} \quad z_j \ge t_j + d_{y_j} - f_{y_j} \quad \text{all } j$$

(job duration, due date)

start time → $$t_{j+1} \ge t_j + d_{y_j} + s_{y_j y_{j+1}} \quad \text{all } j < n$$

(setup time)

$$\text{all-diff}\{y_1, \ldots, y_n\}$$

$$t_1 = 0.$$

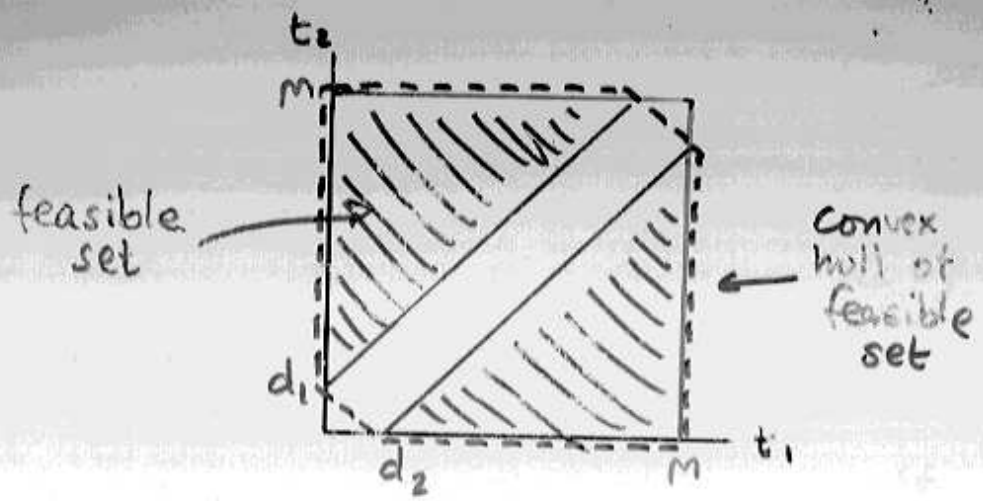
job that occurs 1st in sequence

Typical scheduling constraint

$$(t_2 \geq t_1 + d_1) \vee (t_1 \geq t_2 + d_2)$$

start time of job 1 — duration — or

$0 \leq t_i \leq M$ ← upper bound on makespan



Convex hull relaxation is scarcely better than bounds $0 \leq t_i \leq M$.

Overhead of integer variables not worth it.

Fixed charge network flow problem:

Traditional relaxation –

$$\min \sum_{ij} F_{ij} y_{ij} + c_{ij} x_{ij}$$

$$\text{s.t.} \quad 0 \le x_{ij} \le M y_{ij} \quad \text{all } i,j$$

$$\sum_i x_{ij} - \sum_i x_{ji} = s_j \quad \text{all } j$$

$$0 \le y_{ij} \le 1 \quad \text{all } i,j$$

Projected relaxation (equivalent) –

$$\min \sum_{ij} \left(\frac{F_{ij}}{M} + c_{ij} \right) x_{ij}$$

$$\text{s.t.} \quad 0 \le x_{ij} \le M \quad \text{all } i,j$$

$$\sum_i x_{ij} - \sum_i x_{ji} = s_j \quad \text{all } j$$

This is a min cost network flow problem!

Heavy-duty modeling –

resource-constrained job shop scheduling

$$\min \sum_i z_i$$ ← total tardiness

$$\text{s.t. } z_i \geq t_i + d_i - f_i, \quad \text{all } i$$

$$\sum_k \bar{r}_k \leq R, \quad \text{all } i$$

resource consumed by job k

← resource limit

$$t_k \leq t_i \leq t_k + d_k$$

take snapshot when job i starts

$$t_i \geq 0, \quad \text{all } i$$

Problem: variable index set puts continuous variables t_i in the antecedent

Unpacked version of resource constraint:

$$\sum_k \bar{r}_k \leq R$$

$$(t_k \leq t_i \leq t_k + d_k) \rightarrow (\bar{r}_k = r_k), \quad \text{all } i,k$$

$$\bar{r}_k \geq 0$$

continuous variables in antecedent

Remedy: Use discrete event modeling

$$\min \sum_e z_e$$

← total tardiness

define tardiness ↓

$$\text{s.t. } (U_e = F) \rightarrow (z_e \geq t_e - f_{j_e}) \quad \text{all } e$$

type of event e → U_e

$$(U_e = S) \rightarrow \begin{pmatrix} z_e = 0 \\ \sum_{\substack{e' \leq e \\ U_{e'} = S}} r_{j_{e'}} - \sum_{\substack{e' < e \\ U_{e'} = F}} r_{j_{e'}} \leq R \end{pmatrix}, \quad \text{all } e$$

check resource consumption at start of each job

event that represents start of job j → t_{s_j}

$$t_{F_j} = t_{s_j} + d_j \quad \text{all } j$$

defines S_k, F_k → j_{s_k}, j_{F_k}

$$j_{s_k} = j_{F_k} = k \quad \text{all } k$$

$$t_1 = 0$$

time of event e → t_e

$$t_e \geq t_{e-1} \quad \text{all } e$$

event →

$$e \in \{S, F\}$$

start of job ↑ ↑ finish of job

How about integer variables?

Their 3 traditional functions should be distinguished

- Discrete modeling
- Branching - branch on fractions
- Relaxation -
  - continuous relaxation
  - Lagrangean dual
  - cutting planes

So what should we do?

- Modeling – use logic-based + discrete modeling
  Integer variables can still be the modeling device of choice – e.g. 0-1 knapsack.
- Branching – branch on discrete variables (values need not be integers)
- Relaxation – very important
  - use no relaxation
  - use projection of traditional relaxation onto continuous variable
  - use optimal separating inequalities
  - use relaxation duals, etc.

General Form

$$\min \quad cx$$

$$\text{s.t.} \quad h_i(y) \rightarrow A^i x \geq b^i \quad \text{all } i$$

y discrete

x continuous

Variable subscripts (and index sets) can be reduced to this form ...

e.g. $v_y \leq x \leq v_{y+1}$

can be written

$$(y = i) \rightarrow (v_i \leq x \leq v_{i+1}), \quad \text{all } i$$

but they can also be handled directly in the solver.

How to solve it?

- Branch on discrete variables.
- At each node, solve the LP

Not the LP relaxation →

$$\min\ cx$$
$$\text{s.t. } A^i x \geq b^i, \quad \text{all } i \text{ for which } h_i(y) \text{ is true}$$

otherwise like traditional branch + bound.

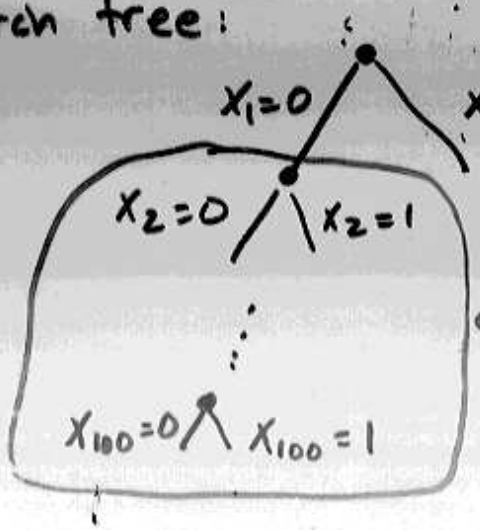
Use logical inference and constraint satisfaction techniques to try to determine if partial assignment to $y_1, \ldots, y_n$ makes $h_i(y)$ true.

In addition, drawing inferences can prune the search tree.

Consider the constraints (0-1)

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

Search tree:



may search subtree of 2^{100} nodes before discovering that $x_1=1$.

Can draw the inference that $x_1=1$.

This makes the constraint set strongly 2-consistent.

This leads to an analog of cutting plane theory.

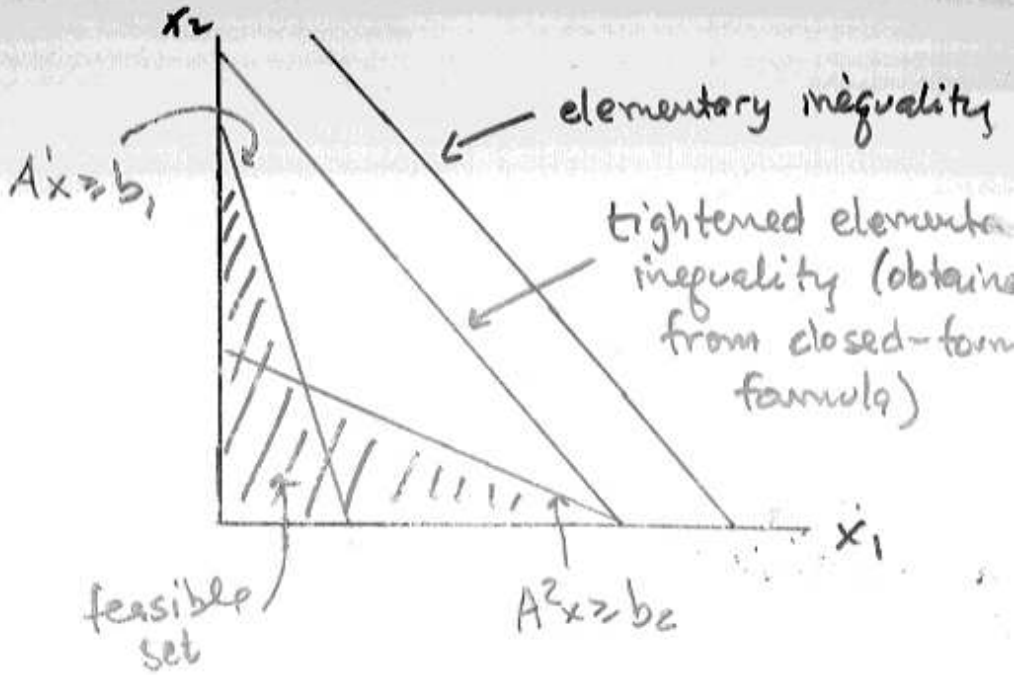
Relaxations for disjunctions —

$$(a^1 x \geq b_1) \vee \cdots \vee (a^m x \geq b_m)$$

Beaumont's elementary inequality:

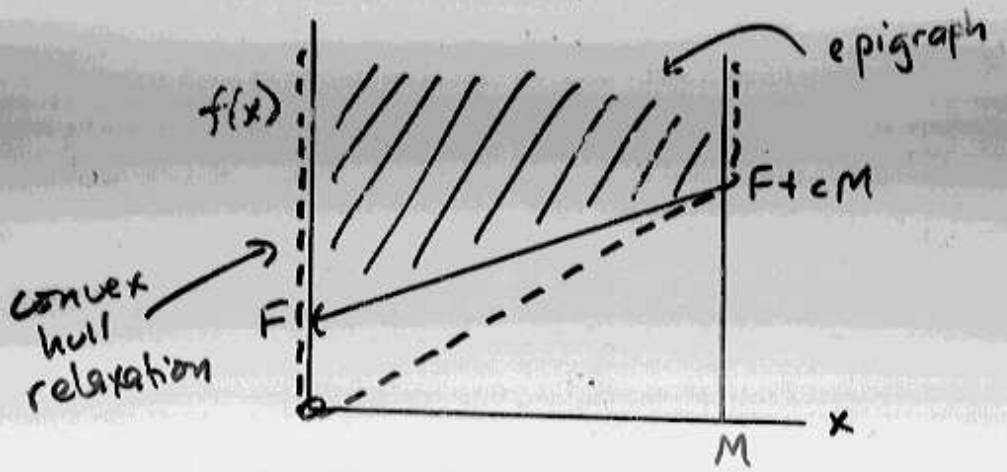
$$\left(\sum_i \frac{a^i}{M_i}\right) x \geq \sum_i \frac{b_i}{M_i} - m + 1$$

Equivalent to projection of traditional big-M relaxation onto x.



Projecting the traditional relaxation onto the continuous variables.

For example, fixed charge problems



$$f(x) = \begin{cases} 0 & \text{if } x = 0 \\ F + cx & \text{if } x > 0 \end{cases}$$

Convex hull relaxation in original continuous variables:

$$f(x) \geq \left(\frac{F}{M} + c\right) x$$

Equivalent to projection of traditional relaxation onto x.

The big-M formulation can have a fractional solution even when the disjunctive constraint is satisfied. So MILP branching continues unnecessarily.

Theorem. In the disjunction

$$(A^1 x \geq b^1) \vee \cdots \vee (A^m x \geq b^m),$$

this can happen iff $y_{ij} > 0$ for some i, j where

$$y_{ij} = \max \; y_i$$

$$\text{s.t.} \quad M_i y_i \leq A^i x - b^i + M_i$$

$$A^j x \geq b^j$$

$$0 \leq x \leq u$$

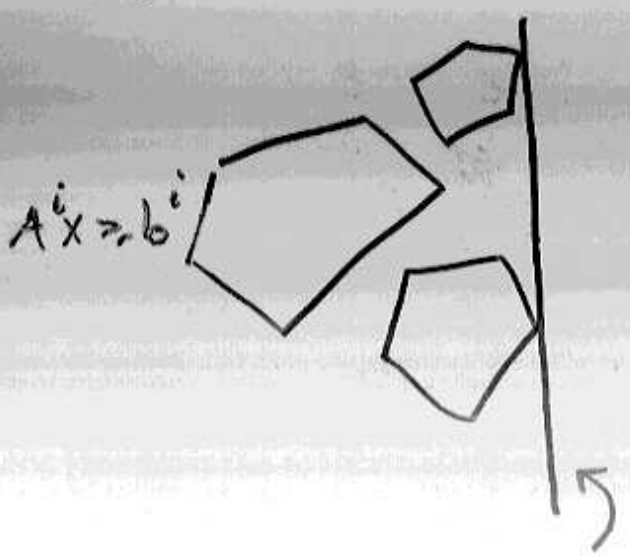
$$y_i \leq 1$$

and $x \leq u$, with $M_i = b^i - \min\{A^i x \mid 0 \leq x \leq u\}$

Optimal Separating Inequalities

Suppose a solution $\bar{x}$ violates the disjunction

$$(A^1 x \geq b^1) \vee \cdots \vee (A^m x \geq b^m).$$



An optimal separating cut $cx \geq r$ is one in which c, r solve

$$\begin{aligned} \max \quad & r - c\bar{x} \\ \text{s.t.} \quad & r \leq u^i b^i \quad \text{all } i \\ & c \geq u^i A^i \quad \text{all } i \\ & -e \leq c \leq e \\ & u^i \geq 0 \end{aligned}$$

When the disjunction is

$$(y_j = 1) \vee (y_j = 0)$$

and the integer variables are retained, this becomes essentially a lift-and-project cut.

Future research ...

Good relaxations for all-diff and other discrete constraints.

Relaxation duals (generalizations of Lagrangean & surrogate duality).

Computational experience –

- Flow shop scheduling with zero-wait transfer. Omit integer variables & get factor of 4 speedup.

- Fixed charge network flow (e.g., chemical process synthesis + warehouse location). No advantage to omitting integer variables if network structure of projected relaxation not used.

- Semicontinuous variables. Omit integer variables + get substantial speedup.

- Progressive party problem. Replace integer variables with discrete variables in logic constraints. Speedup is factor of ≥ 1000.

Link with constraint satisfaction

Recall that the MLLP problem is

$$\begin{aligned} &\min\ cx \\ &\text{s.t. } h_i(y) \rightarrow A^i x \geq b^i \\ &\quad y \text{ discrete} \\ &\quad x \text{ continuous} \end{aligned}$$

Suppose some of the variables $y_1, \ldots, y_n$ have been fixed by branching. Is $h_i(y)$ true?

∴ This is an inference problem – what can one infer from the fixed y_j's?

In particular, can any other y_j's be fixed?

Evaluating $h_i(y)$ sooner creates a better LP bound at the nodes.

Constraint propagation

The game in constraint satisfaction is generally to draw inferences that reduce the domains of discrete variables.

In the process some variables may be fixed.

Resolution is a special case.

A constraint set is strongly k-consistent if whenever values are assigned to $y_1, \ldots, y_{t-1}$ without violating any constraint some value may be assigned to y_t without violating any constraints, and this is true for $t = 1, \ldots, k$.

A constraint set is consistent if it is strongly n-consistent.

If the problem is consistent, a tree search will find a feasible solution without backtracking.

A consistent problem is analogous to an integral polyhedron in MILP.

Resolution is a well-known inference method in propositional logic.

Consistency can be achieved by deriving all possible resolvents (i.e., implications obtained by resolution).

k-consistency can be obtained by deriving all resolvents containing $\leq k$ terms.

Resolvents are analogous to Chvátal cuts in MILP

$$\begin{array}{l} x_1 + x_{100} \geq 1 \\ x_1 - x_{100} \geq 0 \\ \hline x_1 \geq 1 \end{array}$$

resolvent

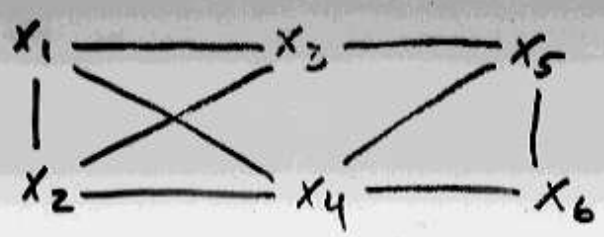
Theorem (Freuder).

A constraint satisfaction problem that is strongly k-consistent can be solved without backtracking if the width of the dependency graph with respect to the branching order is less than k.

Provides a theoretical link between problem structure, constraint generation and branching order.

Dependency graph.

$$
\begin{array}{llll}
x_1 + x_2 + x_3 & & & \geq 1 \\
-x_1 - x_2 + x_3 & & & \geq -1 \\
-x_1 + x_2 & + x_4 & & \geq 0 \\
x_1 - x_2 & + x_4 & & \geq 0 \\
 & x_3 & + x_5 & \geq 1 \\
 & & x_4 + x_5 + x_6 & \geq 1
\end{array}
$$



Width with respect to ordering $1, 2, \ldots, 6$ is 2.

Width with respect to ordering $3, 4, 1, 2, 5, 6$ is 4.

How to structure modeling language to reflect solution options?

Recall general form...

$$\min cx$$
$$\text{st} \quad h_i(y) \rightarrow A^i x \geq b^i \quad \text{all } i$$
$$y \text{ discrete}$$
$$x \text{ continuous}$$

Elements of solution method:

- Enumeration of partial solutions (branching)
- Check whether partial solution makes $h_i(y)$ true (propagation)
- Solve resulting LP (LP solver)

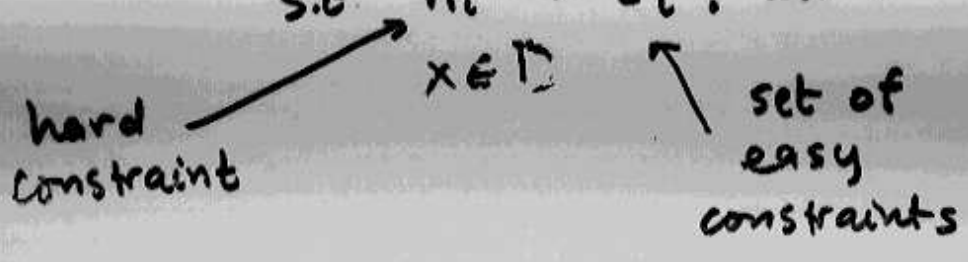
example...

General form of model

$$\min f$$

$$\text{s.t } h_i \rightarrow \bar{E}_i \text{ . all } i$$

$$x \in D$$



- Enumerate test sets (subsets of D defined by easy constraints)
- Check if $x \in$ test set makes h_i true.
- Solve $\bigcup_{i \text{ } h_i \text{ true}} E_i$.

Other realizations of this framework ...

- Enumerate intervals for continuous variables
- Check for constraint satisfaction with interval arithmetic
- Solve constraints in enforced consequents with NLP solver.

This would combine constraint satisfaction and NLP methods for nonlinear optimization

Consider an enumeration of test sets that chooses the next test set in 2 ways: "branch" or "backtrack".

Theorem. If the enumeration never branches on the same test set twice, and never checks a subset of a test set on which it has backtracked, and if the elementary test sets are exhaustive + finite in number, then enumeration solves the test set reformulation.

Conditional constraints provide a means for integrating a checker and a solver within a declarative modeling language.

Roughly speaking,

- T satisfies h_i if all $x \in T$ make h_i true

test set ↗

- T determines h_i if T satisfies h_i or all $x \in T$ make h_i false.

- A test set is elementary if it determines every h_i

- A test set collection is exhaustive if its union = D.

- The value of a test set is min f over that set.

- The test set reformulation is the problem of finding a test set with minimum value in an exhaustive test set collection.