Automating Mathematical Program Transformations

Ashish Agarwal[1*] Sooraj Bhat[2]

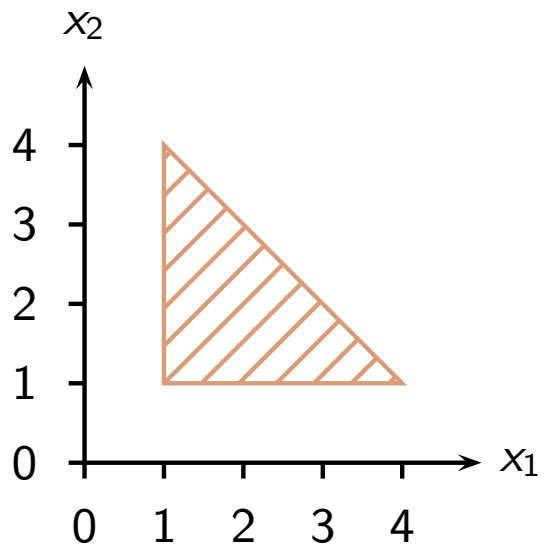
Alexander Gray[2] Ignacio E Grossmann[1]

[1] Carnegie Mellon University
[2] Georgia Institute of Technology
[*] Presently at Yale University

PADL
Madrid, Spain
January 18, 2010

Linear programs (LP)

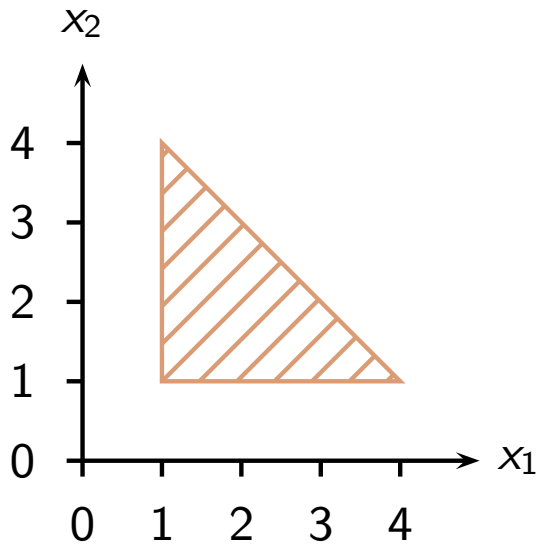
Dantzig (1982)



$$x_1 \geq 1.0$$
$$x_2 \geq 1.0$$
$$x_1 + x_2 \leq 5.0$$

Linear programs (LP)

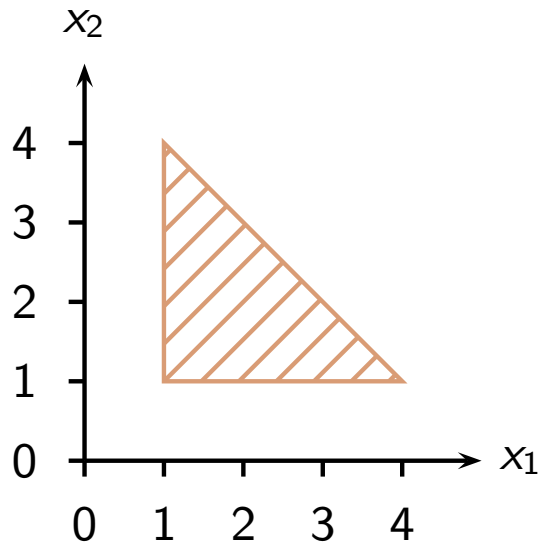
Dantzig (1982)



$$\begin{aligned} \texttt{max}\ & x_1 - x_2 \\ & x_1 \geq 1.0 \\ & x_2 \geq 1.0 \\ & x_1 + x_2 \leq 5.0 \end{aligned}$$

Linear programs (LP)

Dantzig (1982)



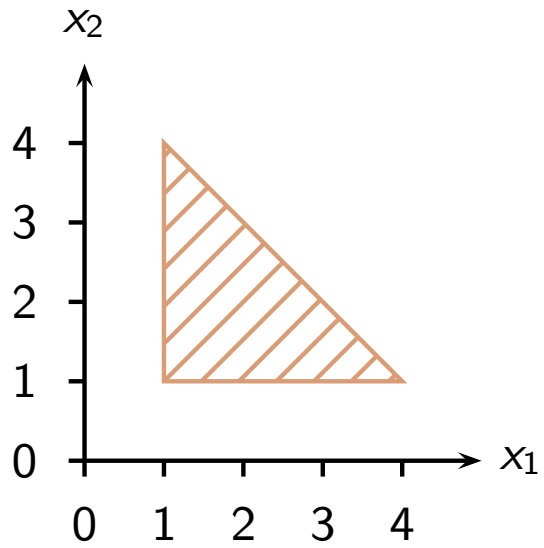
$$\begin{aligned} \max\ & x_1 - x_2 \\ & x_1 \geq 1.0 \\ & x_2 \geq 1.0 \\ & x_1 + x_2 \leq 5.0 \end{aligned}$$

Standard definition:

$\max \left\{ c^T x \mid Ax \leq b, x \in \mathbb{R}^n \right\}$

Linear programs (LP)

Dantzig (1982)



$$\begin{aligned} \max\ & x_1 - x_2 \\ & x_1 \geq 1.0 \\ & x_2 \geq 1.0 \\ & x_1 + x_2 \leq 5.0 \end{aligned}$$

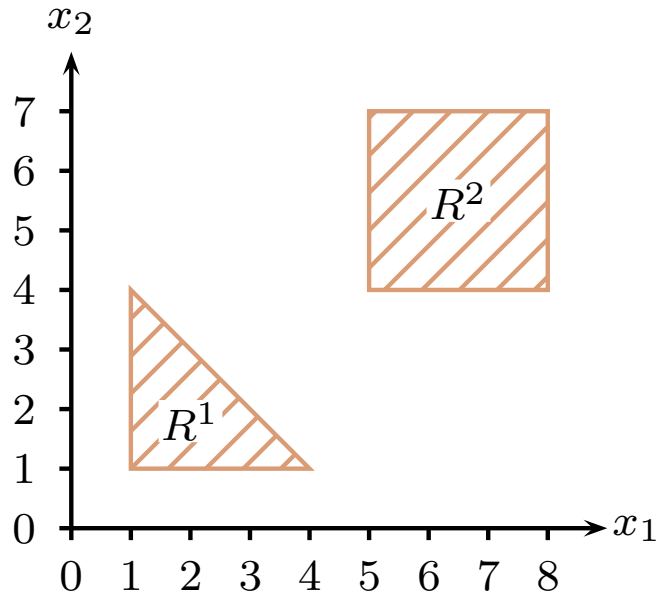
Standard definition:

$\max \left\{ c^T x \mid Ax \leq b, x \in \mathbb{R}^n \right\}$

Cannot represent multiple polyhedra.

Declaring discrete choice – with disjunction

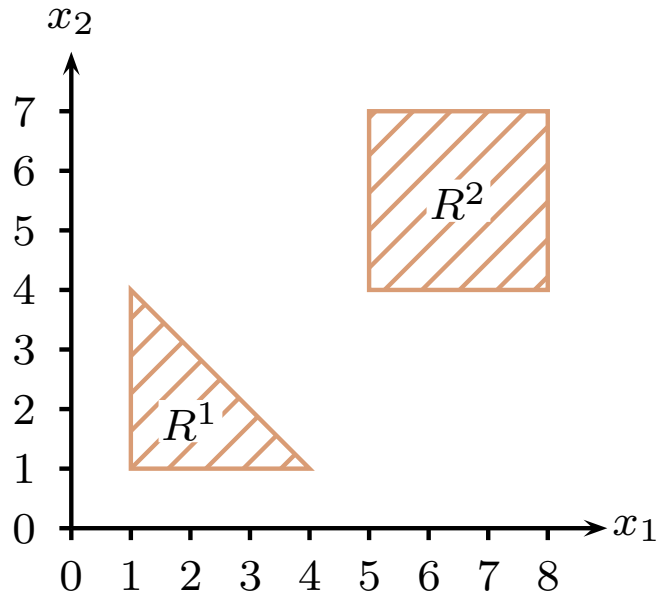
Disjunctive programs (DP), Balas (1974), Jeroslow and Lowe (1984), Raman and Grossmann (1994)



$$\underbrace{\begin{bmatrix} x_1 \geq 1 \\ x_2 \geq 1 \\ x_1 + x_2 \leq 5 \end{bmatrix}}_{R^1} \quad \underbrace{\begin{bmatrix} 5 \leq x_1 \leq 8 \\ 4 \leq x_2 \leq 7 \end{bmatrix}}_{R^2}$$

Declaring discrete choice – with disjunction

Disjunctive programs (DP), Balas (1974), Jeroslow and Lowe (1984), Raman and Grossmann (1994)

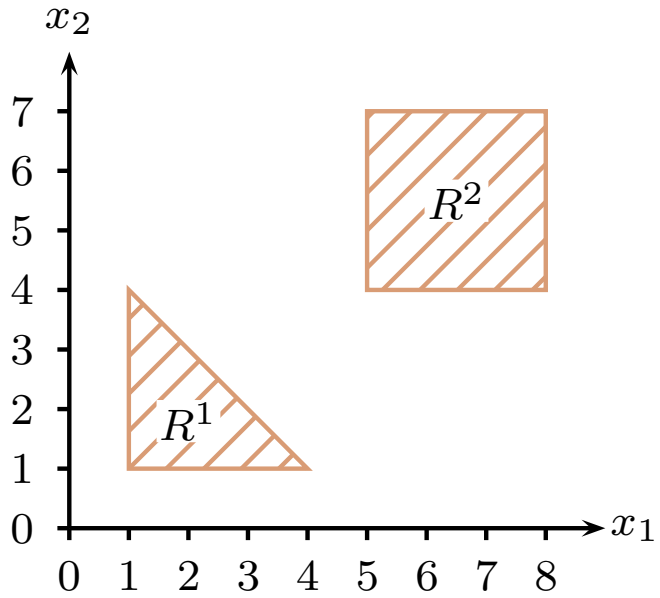


$$\underbrace{\begin{bmatrix} x_1 \geq 1 \\ x_2 \geq 1 \\ x_1 + x_2 \leq 5 \end{bmatrix}}_{R^1} \vee \underbrace{\begin{bmatrix} 5 \leq x_1 \leq 8 \\ 4 \leq x_2 \leq 7 \end{bmatrix}}_{R^2}$$

- Language of DP extends LP with disjunction
- Standard form: $\left[A^1 x \leq b^1\right] \vee \left[A^2 x \leq b^2\right]$

Declaring discrete choice – with disjunction

Disjunctive programs (DP), Balas (1974), Jeroslow and Lowe (1984), Raman and Grossmann (1994)



$$\underbrace{\begin{bmatrix} x_1 \geq 1 \\ x_2 \geq 1 \\ x_1 + x_2 \leq 5 \end{bmatrix}}_{R^1} \vee \underbrace{\begin{bmatrix} 5 \leq x_1 \leq 8 \\ 4 \leq x_2 \leq 7 \end{bmatrix}}_{R^2}$$

- Language of DP extends LP with disjunction
- Standard form: $\left[A^1 x \leq b^1\right] \vee \left[A^2 x \leq b^2\right]$
- Few algorithms for solving DPs directly.

Declaring discrete choice – with integers

Mixed-integer linear programs (MILP)

- Basic idea: multiply terms by $y \in \{0, 1\}$

  $0 \leq y \leq 1$
  $x \leq 3.0y + 2.0(1 - y)$

- if $y = 1$, then $x \leq 3.0$
- if $y = 0$, then $x \leq 2.0$

Declaring discrete choice – with integers

Mixed-integer linear programs (MILP)

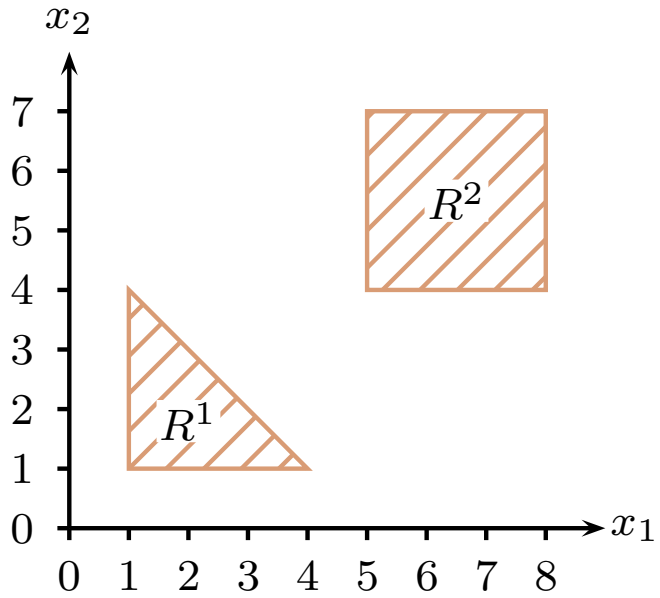
- Basic idea: multiply terms by $y \in \{0, 1\}$

  $0 \leq y \leq 1$
  $x \leq 3.0y + 2.0(1 - y)$

- if $y = 1$, then $x \leq 3.0$
- if $y = 0$, then $x \leq 2.0$
- Language of MILP extends LP with the integer type
- Standard definition: $\max\{c^T x + h^T y \mid Ax + Gy \leq b, x \in \mathbb{R}^n, y \in \mathbb{Z}^p\}$

Declaring discrete choice – with integers

Union of polyhedra

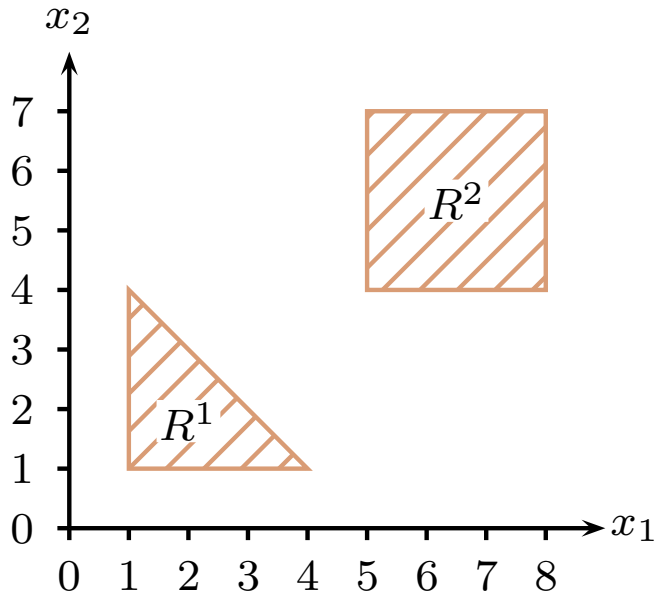


$$\left.\begin{array}{lll} x_1 & \geq 1 & \\ x_2 & \geq 1 & \\ x_1 & + x_2 \quad \leq 5 & \end{array}\right\} R^1$$

$$\left.\begin{array}{lll} 5 & \leq x_1 & \leq 8 \\ 4 & \leq x_2 & \leq 7 \end{array}\right\} R^2$$

Declaring discrete choice – with integers

Union of polyhedra



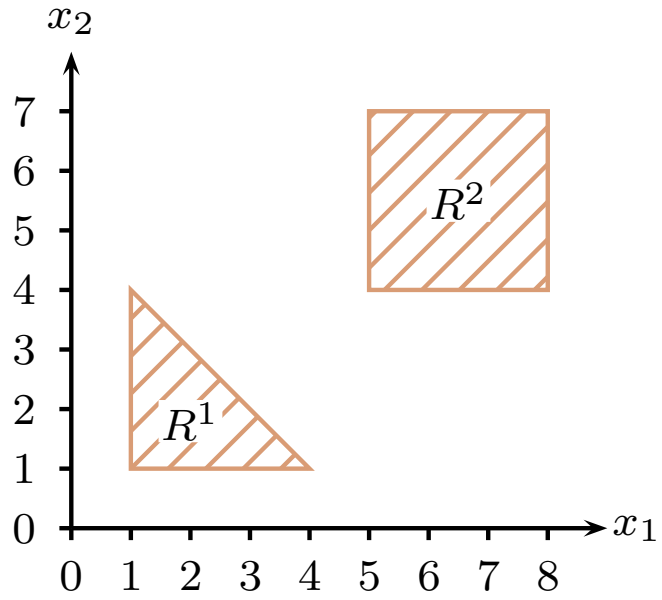
$$\left.\begin{array}{lll} x_1 & \geq 1 & \\ x_2 & \geq 1 & \\ x_1 & + x_2 & \leq 5 \end{array}\right\} R^1$$

$$\left.\begin{array}{lll} 5 & \leq x_1 & \leq 8 \\ 4 & \leq x_2 & \leq 7 \end{array}\right\} R^2$$

$$0 \leq y_1 \leq 1$$
$$0 \leq y_2 \leq 1$$
$$y_1 + y_2 = 1$$

Declaring discrete choice – with integers

Union of polyhedra



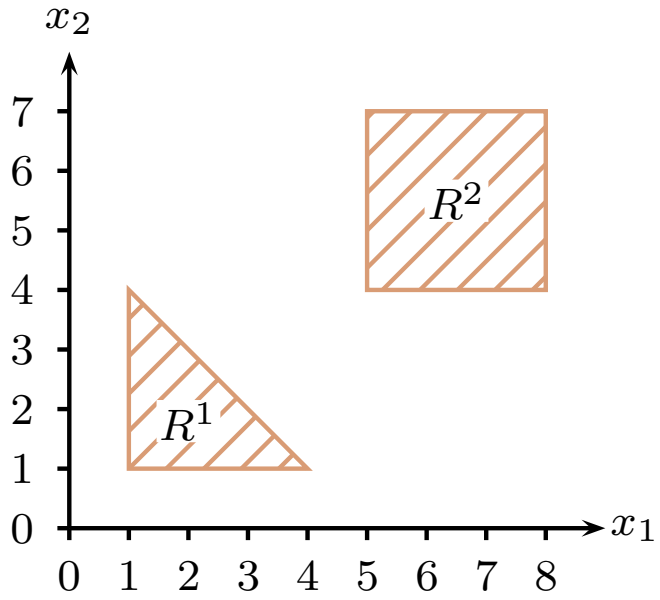
$$\left.\begin{aligned} x_1 y_1 &\geq 1 y_1 \\ x_2 y_1 &\geq 1 y_1 \\ x_1 y_1 + x_2 y_1 &\leq 5 y_1 \end{aligned}\right\} R^1$$

$$\left.\begin{aligned} 5 y_2 \leq x_1 y_2 &\leq 8 y_2 \\ 4 y_2 \leq x_2 y_2 &\leq 7 y_2 \end{aligned}\right\} R^2$$

$$\begin{aligned} 0 \leq y_1 &\leq 1 \\ 0 \leq y_2 &\leq 1 \\ y_1 + y_2 &= 1 \end{aligned}$$

Declaring discrete choice – with integers

Union of polyhedra



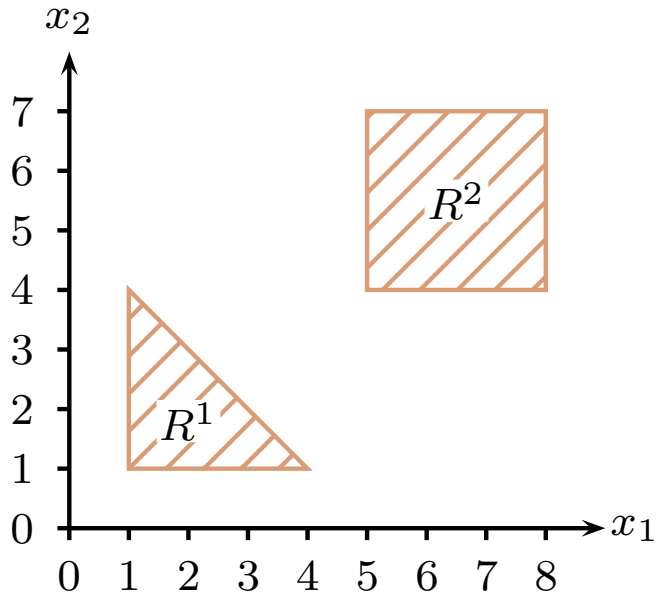
$$\left.\begin{aligned} x_1 y_1 &\geq 1 y_1 \\ x_2 y_1 &\geq 1 y_1 \\ x_1 y_1 + x_2 y_1 &\leq 5 y_1 \end{aligned}\right\} R^1$$

$$\left.\begin{aligned} 5 y_2 \leq x_1 y_2 &\leq 8 y_2 \\ 4 y_2 \leq x_2 y_2 &\leq 7 y_2 \end{aligned}\right\} R^2$$

$$\begin{aligned} 0 \leq y_1 &\leq 1 \\ 0 \leq y_2 &\leq 1 \\ y_1 + y_2 &= 1 \end{aligned}$$

Easy but poor nonlinear formulation.

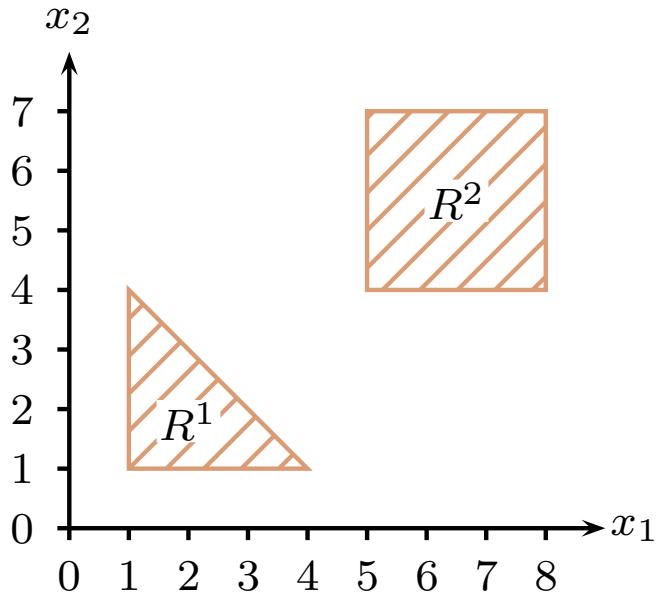
The big-M method



$$\begin{bmatrix} -x_1 + 1 \leq 0 \\ -x_2 + 1 \leq 0 \\ x_1 + x_2 - 5 \leq 0 \end{bmatrix}$$

$$\begin{bmatrix} -x_1 + 5 \leq 0 \\ x_1 - 8 \leq 0 \\ -x_2 + 4 \leq 0 \\ x_2 - 7 \leq 0 \end{bmatrix}$$

The big-M method

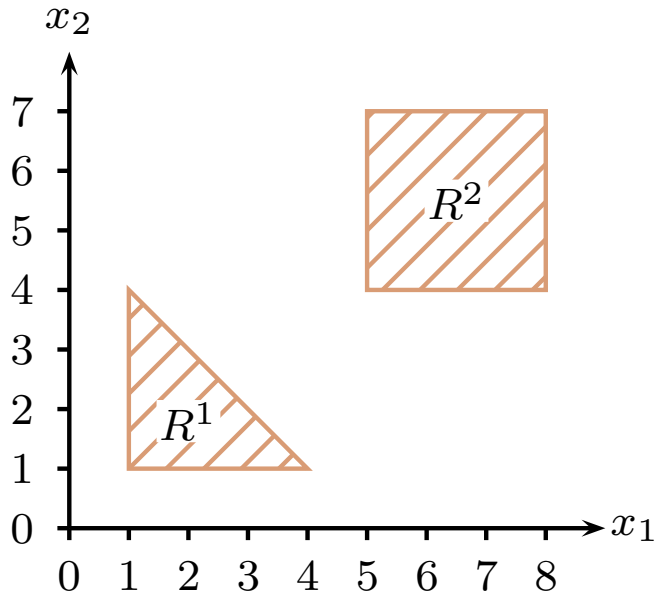


$$\begin{bmatrix} -x_1 + 1 \leq 0 \\ -x_2 + 1 \leq 0 \\ x_1 + x_2 - 5 \leq 0 \end{bmatrix}$$

$$\begin{bmatrix} -x_1 + 5 \leq 0 \\ x_1 - 8 \leq 0 \\ -x_2 + 4 \leq 0 \\ x_2 - 7 \leq 0 \end{bmatrix}$$

$$0 \leq y_1 \leq 1$$
$$0 \leq y_2 \leq 1$$
$$y_1 + y_2 = 1$$
$$M = 10^6$$

The big-M method



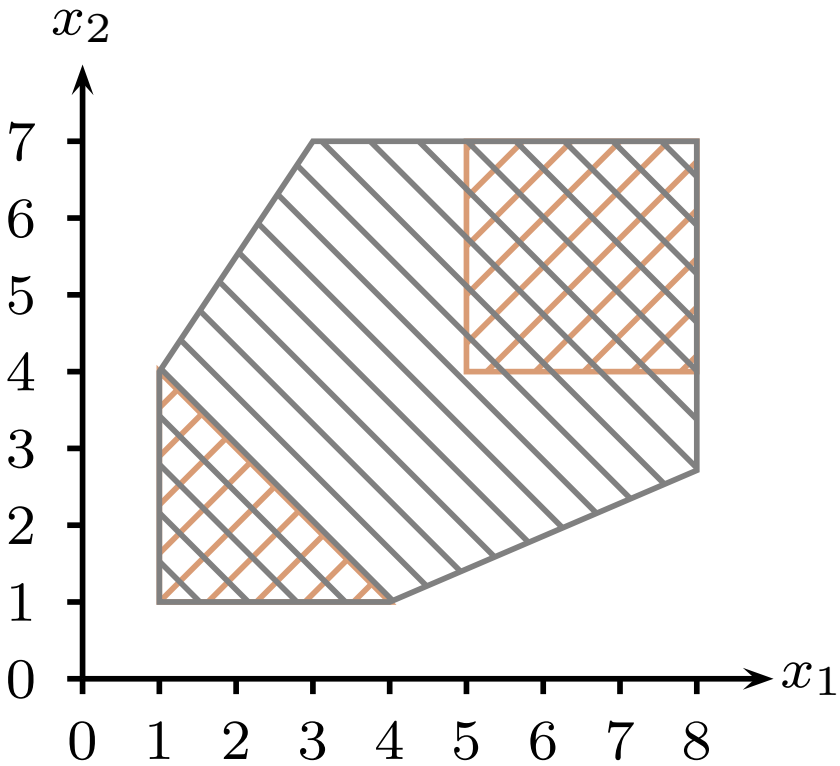
$$\begin{bmatrix} -x_1 + 1 \leq & M(1 - y_1) \\ -x_2 + 1 \leq & M(1 - y_1) \\ x_1 + x_2 - 5 \leq & M(1 - y_1) \end{bmatrix}$$

$$\begin{bmatrix} -x_1 + 5 \leq & M(1 - y_2) \\ x_1 - 8 \leq & M(1 - y_2) \\ -x_2 + 4 \leq & M(1 - y_2) \\ x_2 - 7 \leq & M(1 - y_2) \end{bmatrix}$$

$$0 \leq y_1 \leq 1$$
$$0 \leq y_2 \leq 1$$
$$y_1 + y_2 = 1$$
$$M = 10^6$$

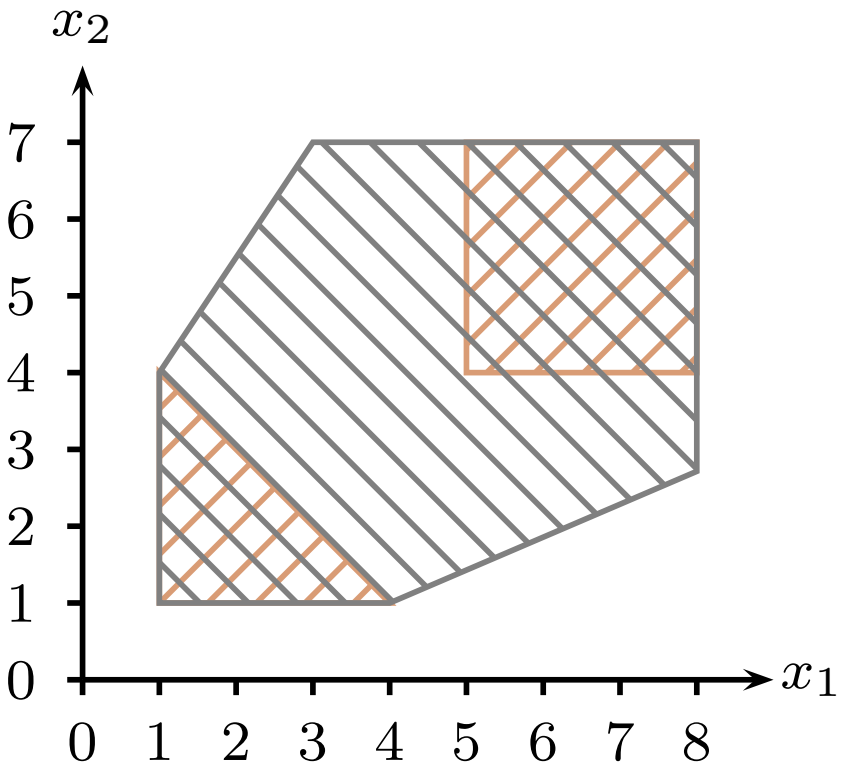
Computational efficiency depends on size of relaxation

best big-M reformulation

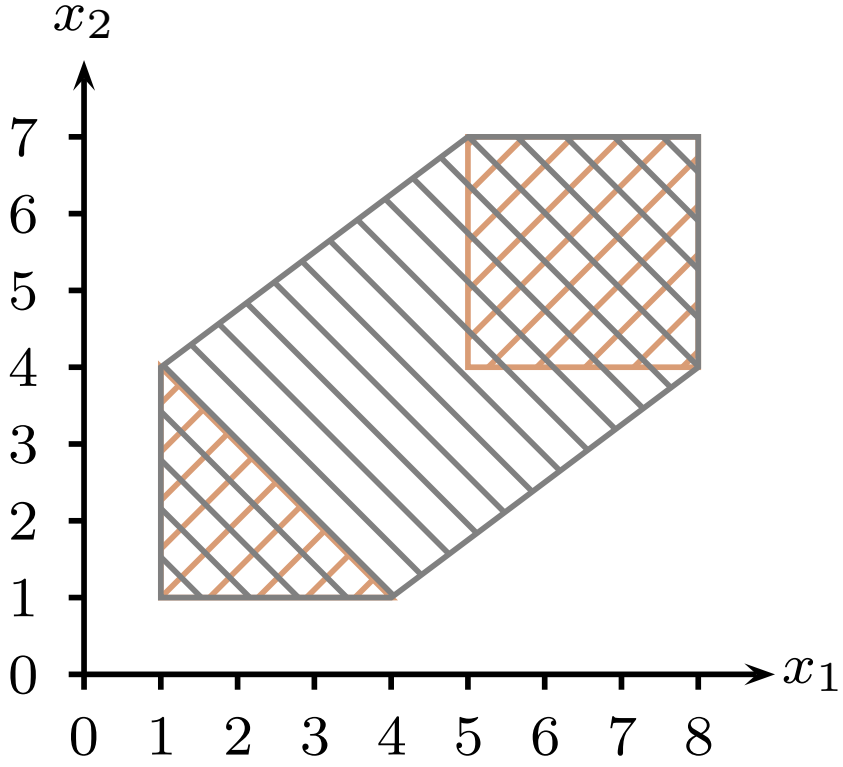


Computational efficiency depends on size of relaxation

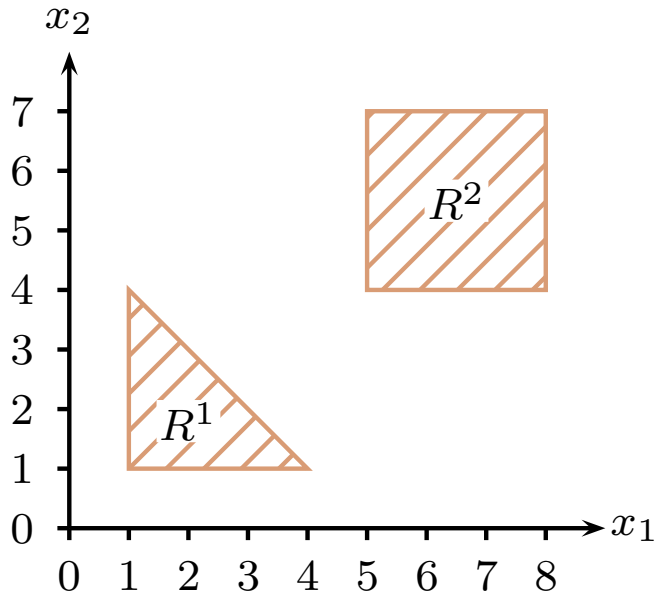
best big-M reformulation



convex-hull reformulation



The convex-hull method

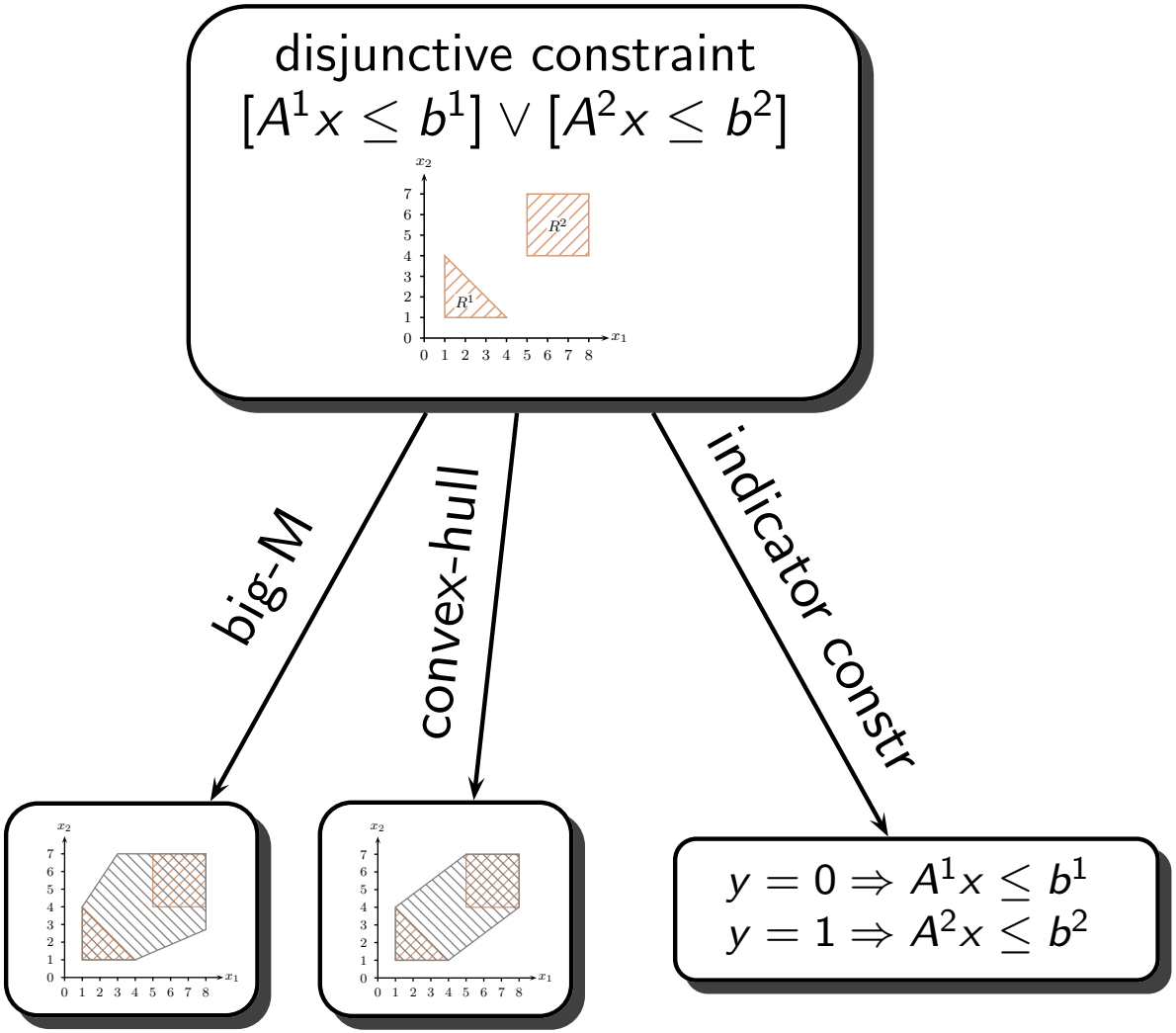


$$\begin{bmatrix} \bar{x}_1^1 \geq 1y_1 \\ \bar{x}_2^1 \geq 1y_1 \\ \bar{x}_1^1 + \bar{x}_2^1 \leq 5y_1 \end{bmatrix}$$

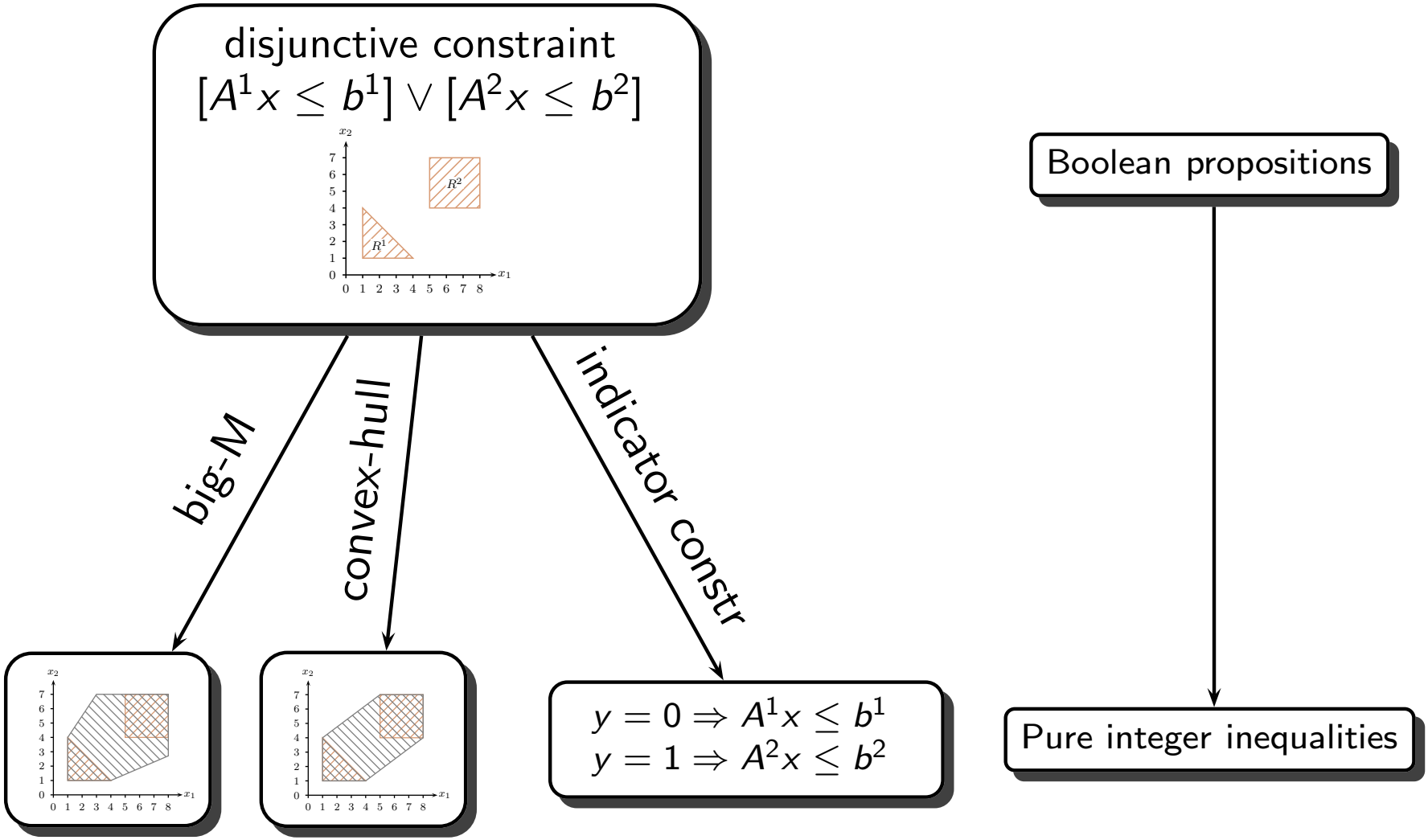
$$\begin{bmatrix} 5y_2 \leq \bar{x}_1^2 \\ \bar{x}_1^2 \leq 8y_2 \\ 4y_2 \leq \bar{x}_2^2 \\ \bar{x}_2^2 \leq 7y_2 \end{bmatrix}$$

$$y_1 + y_2 = 1$$
$$x_1 = \bar{x}_1^1 + \bar{x}_1^2$$
$$x_2 = \bar{x}_2^1 + \bar{x}_2^2$$

Transformations overview



Transformations overview



Previous definition

- Convex-hull reformulation of

$$[A^1 x \leq b^1] \vee [A^2 x \leq b^2]$$

- is

$$\begin{aligned} A^1 \bar{x}^1 &\leq b^1 y_1 & \qquad y_1 + y_2 &= 1 \\ A^2 \bar{x}^2 &\leq b^2 y_2 & \qquad x &= \bar{x}^1 + \bar{x}^2 \end{aligned}$$

Previous definition

- Convex-hull reformulation of

$$[A^1 x \leq b^1] \vee [A^2 x \leq b^2]$$

- is

$$\begin{aligned} A^1 \bar{x}^1 &\leq b^1 y_1 & \qquad y_1 + y_2 &= 1 \\ A^2 \bar{x}^2 &\leq b^2 y_2 & \qquad x &= \bar{x}^1 + \bar{x}^2 \end{aligned}$$

- Definition insufficient for automation
  - Real programs not declared in canonical matrix form
  - Definition limited to numeric features
  - Numerous details omitted:
    - how are variables introduced?
    - disjuncts should be bounded, how is this checked?
    - how are variable bounds tracked?
    - disaggregated variables should have same bounds as those they replace (except range must include zero)

A syntactic foundation for mathematical programs

$$
\begin{aligned}
\rho &::= [r_L, r_U] \mid [r_L, \infty) \mid (-\infty, r_U] \mid \texttt{real} \\
&\quad \mid \langle r_L, r_U \rangle \mid \langle r_L, \infty) \mid (-\infty, r_U \rangle \mid \texttt{int} \\
&\quad \mid \{\texttt{true}\} \mid \{\texttt{false}\} \mid \texttt{bool} \\
e &::= x \mid r \mid \texttt{true} \mid \texttt{false} \mid \texttt{not}\, e \mid e_1\, \texttt{or}\, e_2 \mid e_1\, \texttt{and}\, e_2 \\
&\quad \mid -e \mid e_1 + e_2 \mid e_1 - e_2 \mid e_1 * e_2 \\
c &::= \texttt{T} \mid \texttt{F} \mid \texttt{isTrue}\, e \mid e_1 = e_2 \mid e_1 \leq e_2 \mid c_1 \vee c_2 \mid c_1 \wedge c_2 \mid \exists x : \rho \,.\, c \\
p &::= \max_{x_1:\rho_1,\ldots,x_m:\rho_m} \{e \mid c\} \\
\Upsilon &::= \bullet \mid \Upsilon, x : \rho
\end{aligned}
$$

Types and their refinements

$$\begin{aligned}\rho ::= & [r_L, r_U] \mid [r_L, \infty) \mid (-\infty, r_U] \mid \texttt{real} \\ & \mid \langle r_L, r_U \rangle \mid \langle r_L, \infty) \mid (-\infty, r_U \rangle \mid \texttt{int} \\ & \mid \{\texttt{true}\} \mid \{\texttt{false}\} \mid \texttt{bool}\end{aligned}$$

- `real` and `bool` are the fundamental types
- Numbers treated classically – `int` is subtype of `real`
- Refinements specify upper and/or lower bounds

$$\Upsilon ::= \bullet \mid \Upsilon, x : \rho$$

- More informative than usual type context
- Also maintains restrictions on values a variable can take

Expressions

$$e ::= x \mid r \mid \texttt{true} \mid \texttt{false} \mid \texttt{not}\, e \mid e_1\, \texttt{or}\, e_2 \mid e_1\, \texttt{and}\, e_2$$
$$\mid -e \mid e_1 + e_2 \mid e_1 - e_2 \mid e_1 * e_2$$

- Variables, numeric and Boolean constants
- Boolean and numeric operations

Constraints (Propositions)

$$c ::= \mathtt{T} \mid \mathtt{F} \mid \mathtt{isTrue}\ e \mid e_1 = e_2 \mid e_1 \leq e_2 \mid c_1 \vee c_2 \mid c_1 \wedge c_2 \mid \exists x : \rho\ \mathtt{.}\ c$$

- Equations and inequalities
- Introduce variables with existential quantifier
- Propositional truth `T` and `F` distinct from Boolean `true` and `false`
- Big-M and convex-hull methods apply to $c_1 \vee c_2$, not to e_1 `and` e_2

Programs

$$p ::= \max_{x_1:\rho_1,\ldots,x_m:\rho_m} \{e \mid c\}$$

- Similar to traditional definition except objective and constraint are not in matrix form

big-M transformation $\quad \Upsilon \vdash c \overset{\text{BIGM}}{\longmapsto} c'$

$$\frac{\left\{\Upsilon \vdash c_j \overset{\text{PROP}}{\longmapsto} c'_j\right\}_{j\in\{A,B\}} \qquad \Upsilon \overset{\text{CTXT}}{\longmapsto} \Upsilon' \qquad \left\{\Upsilon' \vdash (1-y_j) \otimes c'_j \multimap c''_j\right\}_{j\in\{A,B\}}}{\Upsilon \vdash c_A \vee c_B \overset{\text{BIGM}}{\longmapsto} \exists y_A : \langle 0,1\rangle \centerdot \exists y_B : \langle 0,1\rangle \centerdot (y_A + y_B = 1) \wedge (c''_A \wedge c''_B)}$$

- Generating the big-M constraint

$$\frac{\Upsilon \vdash e_1 - e_2 \rightleftharpoons [\bar{r}_L, r_U]}{\Upsilon \vdash e \otimes e_1 \leq e_2 \multimap e_1 \leq e_2 + e * r_U}$$

- Computing the big-M parameter

$$\Upsilon \vdash e \rightleftharpoons [\bar{r}_L, \bar{r}_U]$$

Convex-hull transformation $\Upsilon \vdash c \overset{\text{CVX}}{\longmapsto} c'$

$$\frac{\left\{\Upsilon \vdash c_j \overset{\text{PROP}}{\longmapsto} c'_j\right\}_{j\in\{A,B\}} \qquad \Upsilon \overset{\text{CTXT}}{\longmapsto} \Upsilon' \qquad \left\{\Upsilon' \vdash c'_j \multimap_{x_1^j,\ldots,x_m^j} c''_j\right\}_{j\in\{A,B\}} \qquad \left\{y^j \circledast \left\{\vec{x}^j/\vec{x}\right\} c''_j \hookrightarrow c'''_j\right\}_{j\in\{A,B\}}}{\Upsilon \vdash c_A \vee c_B \overset{\text{CVX}}{\longmapsto} \left(\begin{array}{c} \exists \vec{x}^A : \vec{\rho} \,.\, \exists \vec{x}^B : \vec{\rho} \,.\, \exists y^A : \langle 0,1\rangle \,.\, \exists y^B : \langle 0,1\rangle . \\ \left(\vec{x} = \vec{x}^A + \vec{x}^B\right) \wedge \left(y^A + y^B = 1\right) \wedge \left(c'''_A \wedge c'''_B\right) \end{array}\right)}$$

- Compile disjuncts and context
- Add bounding constraints
- Substitute disaggregated variables in each disjunct
- Multiply constant terms by respective y

Example: single disjunctive constraint

Input

```
var x:<10.0, 100.0>
var w:<2.0, 50.0>

min x + w subject_to
(x <= w) disj (x >= w + 4.0)
```

Output

```
var x:<10.0, 100.0>
var w:<2.0, 50.0>

min x + w subject_to

exists y1:[0, 1]
exists y2:[0, 1]
exists x1:<0.0, 100.0>
exists x2:<0.0, 100.0>
exists w1:<0.0, 50.0>
exists w2:<0.0, 50.0>
  w = w1 + w2,
  x = x1 + x2,
  y1 + y2 = 1,

  10.0 * y1 <= x1,
  x1 <= 100.0 * y1,
  2.0 * y1 <= w1,
  w1 <= 50.0 * y1,
  x1 <= w1,

  10.0 * y2 <= x2,
  x2 <= 100.0 * y2,
  2.0 * y2 <= w2,
  w2 <= 50.0 * y2,
  x2 >= w2 + 4.0 * y2
```

Example: single disjunctive constraint

Input

```
var x:<10.0, 100.0>
var w:<2.0, 50.0>

min x + w subject_to
(x <= w) disj (x >= w + 4.0)
```

- Output generated in MPS and AMPL formats
- Implemented as a DSL embedded in OCaml

Output

```
var x:<10.0, 100.0>
var w:<2.0, 50.0>

min x + w subject_to

exists y1:[0, 1]
exists y2:[0, 1]
exists x1:<0.0, 100.0>
exists x2:<0.0, 100.0>
exists w1:<0.0, 50.0>
exists w2:<0.0, 50.0>
  w = w1 + w2,
  x = x1 + x2,
  y1 + y2 = 1,

  10.0 * y1 <= x1,
  x1 <= 100.0 * y1,
  2.0 * y1 <= w1,
  w1 <= 50.0 * y1,
  x1 <= w1,

  10.0 * y2 <= x2,
  x2 <= 100.0 * y2,
  2.0 * y2 <= w2,
  w2 <= 50.0 * y2,
  x2 >= w2 + 4.0 * y2
```

Case studies

Comparison with an automated solution

- switched flow process
- CPLEX Concert: C++ interface to CPLEX
- uses an unknown indicator constraint formulation

Comparison with manual reformulation by an expert

- strip packing
- taken from Sawaya (2006)

All MILPs solved with CPLEX 11.0

Metrics

Number of variables

- continuous
- discrete (binary)

Number of constraints

- numerical relations: $e_1 = e_2$, $e_1 \leq e_2$
- indicator constraints

CPU time required to solve MILP

Metrics

Number of variables

- continuous
- discrete (binary) $\Longleftarrow$

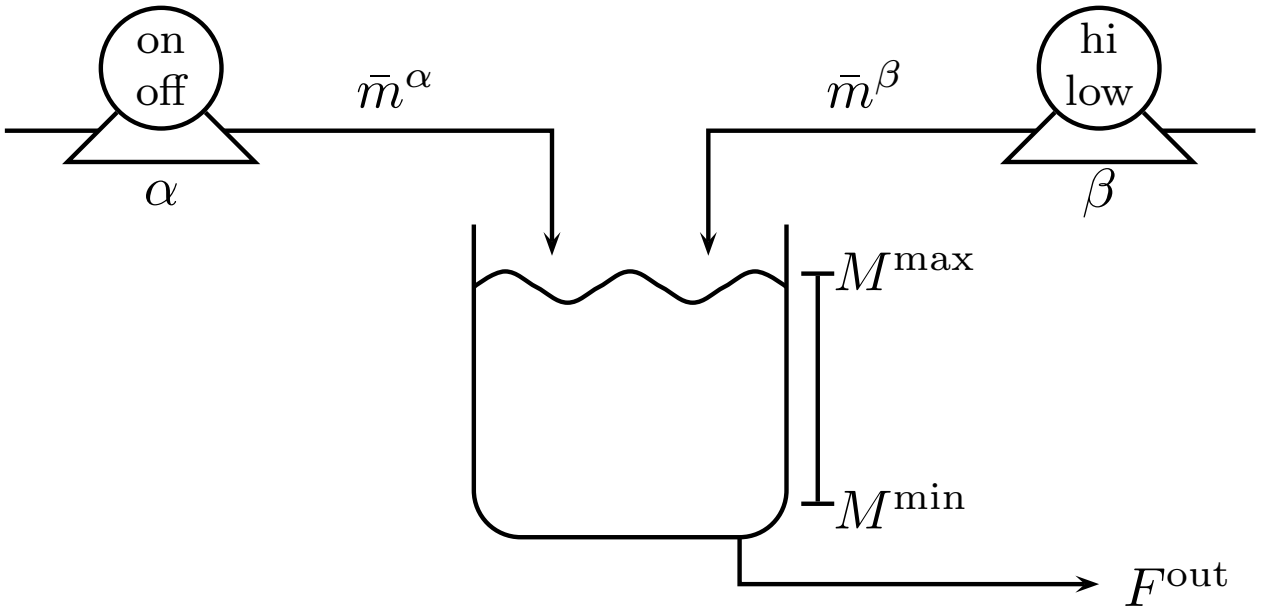
Number of constraints

- numerical relations: $e_1 = e_2$, $e_1 \leq e_2$
- indicator constraints $\Longleftarrow$

CPU time required to solve MILP

- heavily influenced by number of discrete features

Switched flow process



Switched flow process, example constraint

Pump α has three kinds of mode transition dynamics:

$\forall i \in \mathbb{N} \backslash \{n\}$,

$$\left[\begin{array}{c} \text{isTrue } Z^{\alpha}(\text{on}, \text{off}, i) \\ \hat{c}^{\alpha}(i) = 0.0 \\ \hat{r}^{\alpha}(i) = -R^{e}(i) \end{array}\right] \vee \left[\begin{array}{c} \text{isTrue } Z^{\alpha}(\text{off}, \text{on}, i) \\ R^{e}(i) \geq 2.0 \\ \hat{c}^{\alpha}(i) = 50.0 \\ \hat{r}^{\alpha}(i) = -R^{e}(i) \end{array}\right] \vee \left[\begin{array}{c} \text{isTrue } YY^{\alpha}(i) \\ \hat{c}^{\alpha}(i) = 0.0 \\ \hat{r}^{\alpha}(i) = 0.0 \end{array}\right]$$

(Indexed operations are provided at the meta-level.)

Switched flow process, example constraint

Pump α has three kinds of mode transition dynamics:

$\forall i \in \mathbb{N} \backslash \{n\},$

$$\begin{bmatrix} \text{isTrue } Z^{\alpha}(\text{on}, \text{off}, i) \\ \hat{c}^{\alpha}(i) = 0.0 \\ \hat{r}^{\alpha}(i) = -R^{e}(i) \end{bmatrix} \vee \begin{bmatrix} \text{isTrue } Z^{\alpha}(\text{off}, \text{on}, i) \\ R^{e}(i) \geq 2.0 \\ \hat{c}^{\alpha}(i) = 50.0 \\ \hat{r}^{\alpha}(i) = -R^{e}(i) \end{bmatrix} \vee \begin{bmatrix} \text{isTrue } YY^{\alpha}(i) \\ \hat{c}^{\alpha}(i) = 0.0 \\ \hat{r}^{\alpha}(i) = 0.0 \end{bmatrix}$$

...which interact with each other:

$$\forall i \in \mathbb{N} \backslash \{n\}, \forall a \in \{\alpha, \beta\}, \text{ isTrue } YY^{a}(i) \Leftrightarrow \bigvee_{q \in \mathbb{Q}^{a}} Z^{a}(q, q, i)$$

(Indexed operations are provided at the meta-level.)

Switched flow process, example constraint

Pump α has three kinds of mode transition dynamics:

$\forall i \in \mathbb{N}\backslash\{n\}$,

$$\begin{bmatrix} \text{isTrue } Z^{\alpha}(\text{on}, \text{off}, i) \\ \hat{c}^{\alpha}(i) = 0.0 \\ \hat{r}^{\alpha}(i) = -R^{e}(i) \end{bmatrix} \vee \begin{bmatrix} \text{isTrue } Z^{\alpha}(\text{off}, \text{on}, i) \\ R^{e}(i) \geq 2.0 \\ \hat{c}^{\alpha}(i) = 50.0 \\ \hat{r}^{\alpha}(i) = -R^{e}(i) \end{bmatrix} \vee \begin{bmatrix} \text{isTrue } YY^{\alpha}(i) \\ \hat{c}^{\alpha}(i) = 0.0 \\ \hat{r}^{\alpha}(i) = 0.0 \end{bmatrix}$$

...which interact with each other:

$$\forall i \in \mathbb{N}\backslash\{n\}, \forall a \in \{\alpha, \beta\}, \text{ isTrue } YY^{a}(i) \Leftrightarrow \bigvee_{q \in \mathbb{Q}^{a}} Z^{a}(q, q, i)$$

Booleans and disjunction enable the natural modeling of such logical relations between constraints.

(Indexed operations are provided at the meta-level.)

Switched flow process, metrics

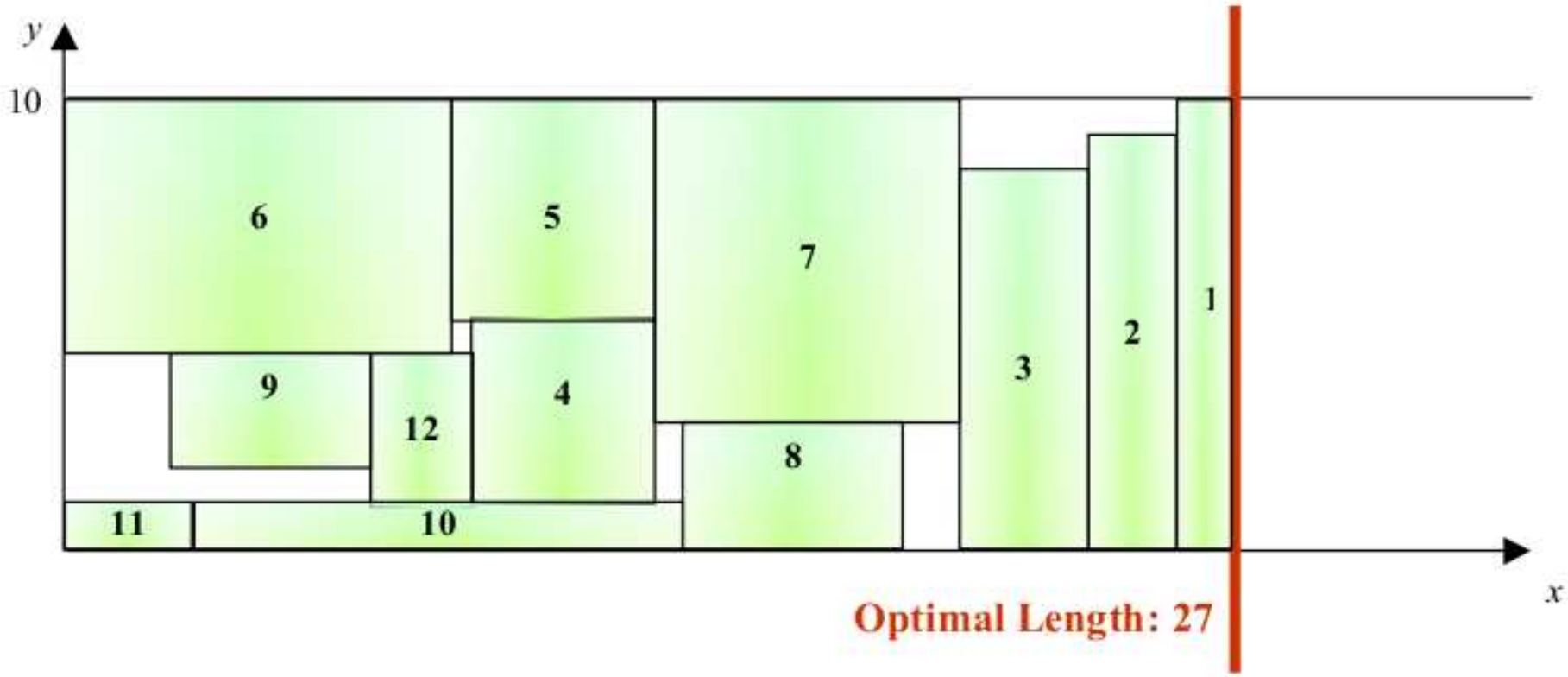
Method	#vars (#binary)	#constr. (#IC)	time (sec)
flow-Concert	1061 (874)	1080 (718)	36.85
flow-IC	477 (291)	1001 (438)	11.60
flow-BM	477 (291)	1198	3.37
flow-CH	1194 (631)	2747	1.09

Switched flow process, metrics

Method	#vars (#binary)	#constr. (#IC)	time (sec)
flow-Concert	1061 (874)	1080 (718)	36.85
flow-IC	477 (291)	1001 (438)	11.60
flow-BM	477 (291)	1198	3.37
flow-CH	1194 (631)	2747	1.09

- The tightest reformulation wins, even with more constraints.

Strip packing



Strip packing, formulation

The most natural formulation uses disjunction.

$$
\begin{aligned}
\min \quad & \mathit{length} \\
\text{s.t.} \quad & \mathit{length} \geq x_i + L_i \quad \forall i \in \mathbb{N} \\
\mathit{no\ overlapping} \quad & \begin{cases} [x_i + L_i \leq x_j] \\ \vee\ [x_j + L_j \leq x_i] \\ \vee\ [y_i - H_i \geq y_j] \\ \vee\ [y_j - H_j \geq y_i] \end{cases} \quad \forall i, j \in \mathbb{N}, i < j \\
\mathit{stay\ in\ bounds} \quad & \begin{cases} 0 \leq x_i \leq L_{\max} - L_i & \forall i \in \mathbb{N} \\ H_i \leq y_i \leq W & \forall i \in \mathbb{N} \end{cases}
\end{aligned}
$$

(x_i, y_i) is the position of the top-left corner of rectangle i.

Strip packing, metrics

Method	#vars (#binary)	#constr. (#IC)	time (sec)
pack12-IC	289 (264)	342 (264)	1.83
pack12-BM	289 (264)	342	1.22
pack12-CH	1345 (264)	2718	168.38
pack12-BM-expert	289 (264)	342	1.82
pack12-CH-expert	1345 (264)	1662	149.57
pack21-IC	883 (840)	1071 (840)	24.44
pack21-BM	883 (840)	1071	55.01
pack21-CH	4243 (840)	8631	991.68
pack21-BM-expert	883 (840)	1071	29.56
pack21-CH-expert	4243 (840)	5271	$\geq$ 3600.00

Strip packing, metrics

Method	#vars (#binary)	#constr. (#IC)	time (sec)
pack12-IC	289 (264)	342 (264)	1.83
pack12-BM	289 (264)	342	1.22
pack12-CH	1345 (264)	2718	168.38
pack12-BM-expert	289 (264)	342	1.82
pack12-CH-expert	1345 (264)	1662	149.57
pack21-IC	883 (840)	1071 (840)	24.44
pack21-BM	883 (840)	1071	55.01
pack21-CH	4243 (840)	8631	991.68
pack21-BM-expert	883 (840)	1071	29.56
pack21-CH-expert	4243 (840)	5271	$\geq$ 3600.00

- Our mechanizations perform just as well as expert encodings.

Strip packing, metrics

Method	#vars (#binary)	#constr. (#IC)	time (sec)
pack12-IC	289 (264)	342 (264)	1.83
pack12-BM	289 (264)	342	1.22
pack12-CH	1345 (264)	2718	168.38
pack12-BM-expert	289 (264)	342	1.82
pack12-CH-expert	1345 (264)	1662	149.57
pack21-IC	883 (840)	1071 (840)	24.44
pack21-BM	883 (840)	1071	55.01
pack21-CH	4243 (840)	8631	991.68
pack21-BM-expert	883 (840)	1071	29.56
pack21-CH-expert	4243 (840)	5271	$\geq$ 3600.00

- Our mechanizations perform just as well as expert encodings.
- No single best reformulation for all cases.

Conclusions

- Automated bigM and convex-hull methods
  None of GAMS, AMPL, Mosel, ILOG provide this
- Framework for mechanizing additional operations: e.g. inserting cuts
- Automation opens the way for *compiler optimizations*

- Provided enriched language for declaring MPs
- Our current work includes probabilistic extensions