Mechanizing Optimization and Statistics

Ashish Agarwal

Yale University

IBM Programming Languages Day
Watson Research Center
July 29, 2010

Acknowledgments

- Optimization:

  Ignacio Grossmann (Carnegie Mellon)
  Nick Sawaya and Vikas Goel (Exxon Mobil)

- Indexing:

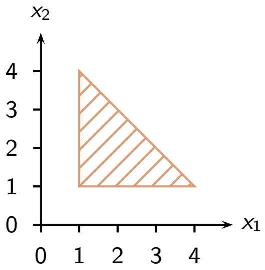
  Bob Harper (Carnegie Mellon)

- Statistics:

  Sooraj Bhat and Alex Gray (GeorgiaTech)
  Rich Vuduc (GeorgiaTech, linear algebra and autotuning)

Linear programs (LP)

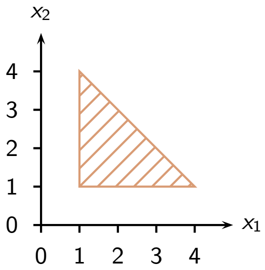
Dantzig (1982)



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

Linear programs (LP)

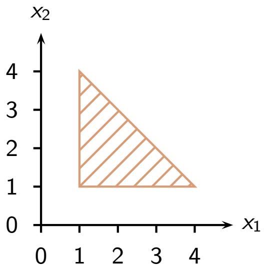
Dantzig (1982)



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

Linear programs (LP)

Dantzig (1982)

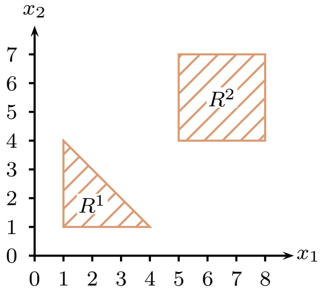


$$
\begin{aligned}
\texttt{max}\ & x_1 - x_2 \\
& x_1 \geq 1.0 \\
& x_2 \geq 1.0 \\
& x_1 + x_2 \leq 5.0
\end{aligned}
$$

Cannot represent multiple polyhedra.

Declaring discrete choice – with disjunction

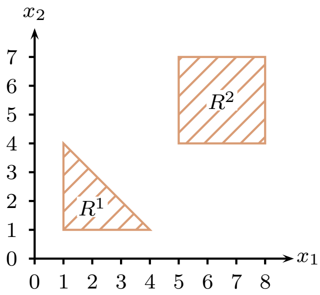
Disjunctive programs (DP), Balas (1974), Jeroslow and Lowe (1984), Raman and Grossmann (1994)



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

Declaring discrete choice – with disjunction

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

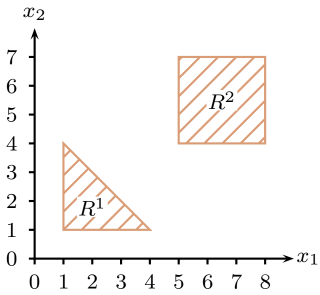


$$\underbrace{\begin{bmatrix} x_1 \geq 1 \\ x_2 \geq 1 \\ x_1 + x_2 \leq 5 \end{bmatrix}}_{R^1} \vee \underbrace{\begin{bmatrix} 5 \leq x_1 \leq 8 \\ 4 \leq x_2 \leq 7 \end{bmatrix}}_{R^2}$$

- Language of DP extends LP with disjunction

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
- Few algorithms for solving DPs directly.

Declaring discrete choice – with integers

Mixed-integer linear programs (MILP)

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

$$
\begin{aligned}
&0 \le y \le 1 \\
&x \le 3.0y + 2.0(1 - y)
\end{aligned}
$$

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

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

Declaring discrete choice – with integers

Mixed-integer linear programs (MILP)

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

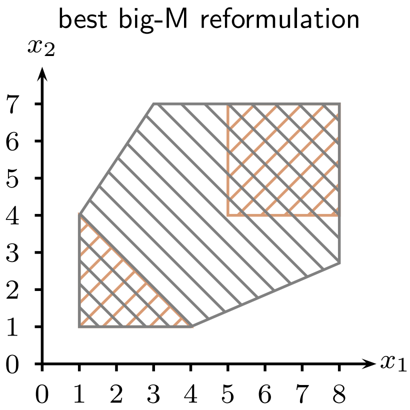
$$0 \leq y \leq 1$$
$$x \leq 3.0y + 2.0(1 - y)$$

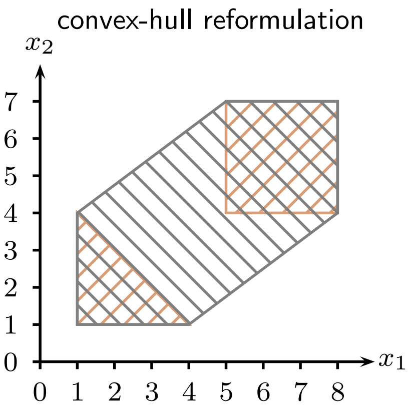
- if $y = 1$, then $x \leq 3.0$
- if $y = 0$, then $x \leq 2.0$
- Language of MILP extends LP with the integer type

Goal: Express as DP (intuitive) $\longrightarrow$ Convert to MILP (accepted by solvers)

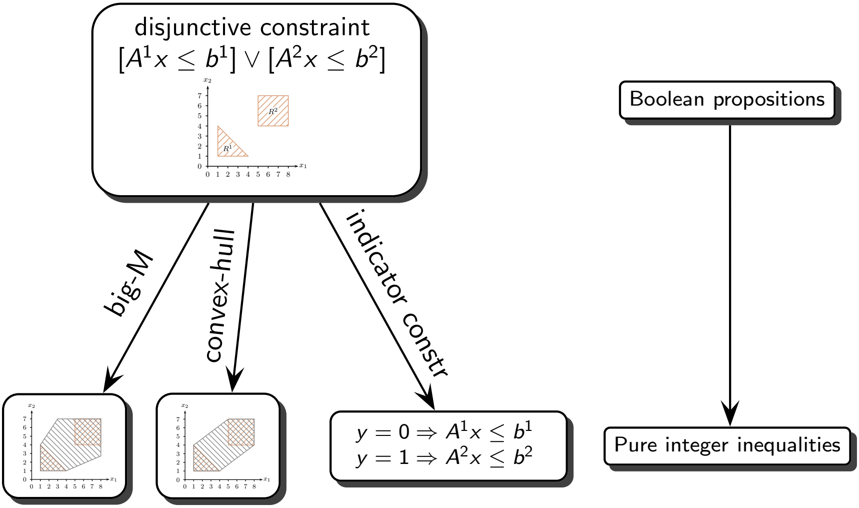
Multiple Transformation Techniques Available

Choice affects computational efficiency





System 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

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

- Insufficient for automation
    - Real programs not declared in canonical matrix form
    - 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

Agarwal, Bhat, Gray, Grossmann (2010)

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

Convex-hull transformation $\quad \Upsilon \vdash c \xmapsto{\text{CVX}} c'$

Agarwal, Bhat, Gray, Grossmann (2010)

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

- 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

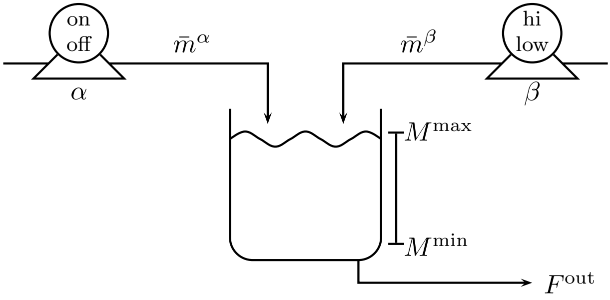
```
var x:<10.0, 100.0>
var w:<2.0, 50.0>

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
```

Switched flow process



Switched flow process, example constraint

Pump α has three kinds of mode transition dynamics:

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

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

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

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.

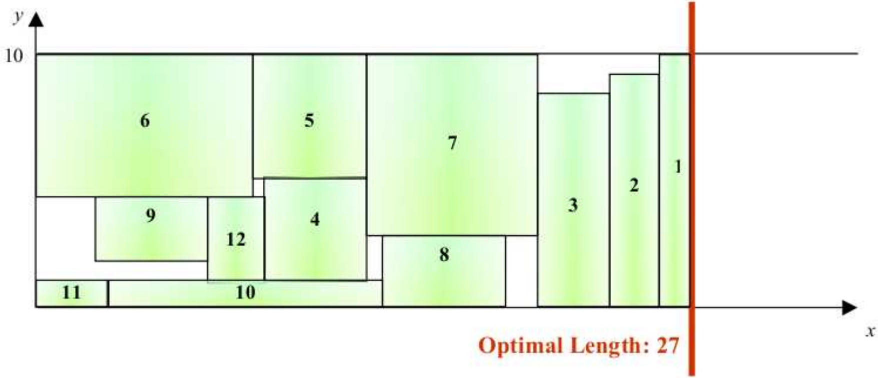
Method	#vars (#binary)	#constr. (#IC)	time (sec)
flow-Concert	1061 (874)	1080 (718)	36.85
flow-IC	477 (291)	1001 (438)	11.60
flow-BM	477 (291)	1198	3.37
flow-CH	1194 (631)	2747	1.09

Switched flow process, comparison to ILOG Concert

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

- All 3 of our methods improve on state-of-the-art.

Strip packing



Strip packing, formulation

The most natural formulation uses disjunction.

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

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

Strip packing, comparison to expert

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	$>$ 3600.00

Strip packing, comparison to expert

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	$>$ 3600.00

- Our mechanizations perform just as well as expert encodings.

Indexing is Ubiquitous

- We solved a problem with 150,000 equations and 25,000 variables.
- How were so many equations and variables declared?

Indexing is Ubiquitous

- We solved a problem with 150,000 equations and 25,000 variables.
- How were so many equations and variables declared?
  Sometimes, use matrix notation. Often, use indexing.

Indexing is Ubiquitous

- We solved a problem with 150,000 equations and 25,000 variables.
- How were so many equations and variables declared?
  Sometimes, use matrix notation. Often, use indexing.
- Indexed operators:

$$\sum_{i=1}^{n} x_i$$

- Families of equations:

$$\forall i \in [1 \ldots n] \centerdot x_{i+1} = x_i + y_i$$

- Indexing is a meta-programming feature
- Index variables i distinct from mathematical variables x

Complex Index Sets Arise in Real Problems

- Job shop scheduling:

$$\forall j \in J \,.\, \forall s \in S_j \,.\, \forall j' \in Pre_{j,s} \,.\, t_{j',s} \leq t_{j,s}$$

- Mappings from sets to set of all sets: S
- Dependent types: S_j depends on value of j

Indexing Language: Syntax

Agarwal (2006)

- Index Expressions

$$
\begin{aligned}
\varepsilon ::= {} & i \mid k \\
& \mid (\varepsilon_1, \ldots, \varepsilon_m) \mid \varepsilon.k \\
& \mid -\varepsilon \mid \varepsilon_1 + \varepsilon_2 \mid \varepsilon_1 - \varepsilon_2 \mid \varepsilon_1 * \varepsilon_2 \\
& \mid \texttt{case}\ \varepsilon\ \texttt{of}\ \{k_j \Rightarrow \varepsilon_j\}_{j=1}^{m}
\end{aligned}
$$

- Index Sets (Types)

$$
\begin{aligned}
\sigma ::= {} & [\varepsilon_L..\varepsilon_U] \mid i_1 : \sigma_1 \times \cdots \times i_m : \sigma_m \\
& \mid \texttt{case}\ \varepsilon\ \texttt{of}\ \{k_j \Rightarrow \sigma_j\}_{j=1}^{m} \\
& \mid \lambda i \centerdot \sigma \mid \sigma\, \varepsilon \\
& \mid \sigma :: \kappa
\end{aligned}
$$

- Kinds

$$\kappa ::= \texttt{IndexSet} \mid i : \sigma \Rightarrow \kappa$$

Example Index Sets

```
set JOBS = {'a','b','c'}

set STAGES = fn i . case i of
                      'a' => {'s1','s2'}
                    | 'b' => {'s1','s3','s4'}
                    | 'c' => {'s3','s4'}

set JOBS_STAGES = i:JOBS * STAGES[i]
```

Explicitly:

```
{('a','s1'), ('a','s2'),
 ('b','s1'), ('b','s3'), ('b','s4'),
 ('c','s3'), ('c','s4')}
```

Memory Reduction

- Load this program:

$$\forall i \in [1 \ldots n] \,.\, x_{i+1} = x_i + y_i$$

- Other software expand this to:

$$\begin{aligned} x_2 &= x_1 + y_1 \\ x_3 &= x_2 + y_2 \\ x_4 &= x_3 + y_3 \\ x_5 &= x_4 + y_4 \\ \vdots \; & \quad \vdots \quad\quad \vdots \end{aligned}$$

Memory Reduction

- Load this program:

$$\forall i \in [1 \ldots n] \,.\, x_{i+1} = x_i + y_i$$

- Other software expand this to:

$$\begin{aligned} x_2 &= x_1 + y_1 \\ x_3 &= x_2 + y_2 \\ x_4 &= x_3 + y_3 \\ x_5 &= x_4 + y_4 \\ \vdots \quad & \quad \vdots \quad \quad \vdots \end{aligned}$$

- We retain indexing structure:
  Memory requirements reduced from $O(n)$ to $O(1)$.

Computational Improvements

- Input to our software:

$$\bigvee_{i:[1..10]} w \geq x_i + 4.0$$

- Our software's output:

$$\bigwedge_{i:[1..10]} \begin{bmatrix} 10.0 * y_i \leq w'_i \\ w'_i \leq 90.0 * y_i \\ \bigwedge_{d:[1..10]} \begin{bmatrix} 5.0 * y_i \leq x'_{i,d} \\ x'_{i,d} \leq 75.0 * y_i \end{bmatrix} \\ w'_i \geq x'_{i,i} + 4.0 * y_i \end{bmatrix}$$

Computational Improvements

- Input to our software:

$$\bigvee_{i:[1..10]} w \geq x_i + 4.0$$

- Our software's output:

$$\bigwedge_{i:[1..10]} \begin{bmatrix} 10.0 * y_i \leq w'_i \\ w'_i \leq 90.0 * y_i \\ \bigwedge_{d:[1..10]} \begin{bmatrix} 5.0 * y_i \leq x'_{i,d} \\ x'_{i,d} \leq 75.0 * y_i \end{bmatrix} \\ w'_i \geq x'_{i,i} + 4.0 * y_i \end{bmatrix}$$

Reformulation time reduced from $O(n)$ to $O(1)$.

Indexing Language: Type System and Semantics

Agarwal (2006)

- Usual Way: Syntax $\longrightarrow$ Type System $\longrightarrow$ Semantics
  Existence is prior to meaning.
- Alternative Way: Syntax $\longrightarrow$ Semantics $\longrightarrow$ Type System
  Meaning is prior to existence.

Indexing Language: Type System and Semantics

Agarwal (2006)

- Usual Way: Syntax $\longrightarrow$ Type System $\longrightarrow$ Semantics
  Existence is prior to meaning.
- Alternative Way: Syntax $\longrightarrow$ Semantics $\longrightarrow$ Type System
  Meaning is prior to existence.

Admits more programs. Possible only because all types are finitary.

What Are Random Variables?

- Wasserman (2004) says:

  A random variable is a mapping

  $$X : \Omega \to \mathbb{R}$$

  that assigns a real number $X(\omega)$ *to each outcome* ω.

What Are Random Variables?

- Wasserman (2004) says:

  A random variable is a mapping
  $$X : \Omega \to \mathbb{R}$$
  that assigns a real number $X(\omega)$ *to each outcome* ω.

However:

- Treated as real: $\mathbb{P}(X \geq 5)$
- Not random:
  We write
  $$X \sim \texttt{Bernoulli}(p)$$
  to mean that X is *exactly* distributed as
  $$f(x) = p^x(1-p)^{1-x} \text{ for } x \in \{0, 1\}$$

What Are Random Variables?

Not variables:

- Cannot substitute occurrences of X for anything.
  e.g. In $\mathbb{P}(X \geq 5)$, certainly cannot replace X with its distribution.
- Dependence matters.
  e.g. Two random variables X and Y, both distributed as `Bernoulli`(0.5), each 0 or 1 with probability 0.5. What is $\mathbb{P}(X + Y = 2)$?

  Perhaps 0.25? But not if $Y = 1 - X$.

What Are Random Variables?

Not variables:

- Cannot substitute occurrences of X for anything.
  e.g. In $\mathbb{P}(X \geq 5)$, certainly cannot replace X with its distribution.
- Dependence matters.
  e.g. Two random variables X and Y, both distributed as `Bernoulli`(0.5), each 0 or 1 with probability 0.5. What is $\mathbb{P}(X + Y = 2)$?

  Perhaps 0.25? But not if $Y = 1 - X$.

Random variables are neither random nor variable.

Previous Work

- Giry (1981), Jones and Plotkin (1989)
  Probability distributions are a monad.
- Kozen (1981)
  Formalized semantics.
- Ramsey and Pfeffer (2002)
  Efficient expectations, but discrete distributions only.
- Park, Pfenning, and Thrun (2004)
  Continuous distributions also, but support only sampling.
- Erwig and Kollmansberger (2006)
  Provide Haskell library, but discrete distributions only, computational efficiency not optimized.

Previous Work

- Giry (1981), Jones and Plotkin (1989)
  Probability distributions are a monad.
- Kozen (1981)
  Formalized semantics.
- Ramsey and Pfeffer (2002)
  Efficient expectations, but discrete distributions only.
- Park, Pfenning, and Thrun (2004)
  Continuous distributions also, but support only sampling.
- Erwig and Kollmansberger (2006)
  Provide Haskell library, but discrete distributions only, computational efficiency not optimized.

Our goal: Unify these results in a single system.

Syntax: Probability Language

Bhat, Agarwal, Gray, Vuduc (2010)

$$
\begin{aligned}
T &::= \texttt{Bool} \mid \texttt{Int} \mid \texttt{Real} \mid T_1 \times T_2 \mid \texttt{Prob}\ T \\
E &::= x \mid \texttt{true} \mid \texttt{false} \\
&\quad \mid r \mid E_1 + E_2 \mid E_1 \times E_2 \\
&\quad \mid (E_1, E_2) \mid \texttt{fst}\ E \mid \texttt{snd}\ E \\
&\quad \mid \texttt{if}\ E_1\ \texttt{then}\ E_2\ \texttt{else}\ E_3 \mid E_1 = E_2 \mid E_1 \leq E_2 \\
&\quad \mid \texttt{uniform} \mid \texttt{return}\ E \mid \texttt{let}\ x \sim E_1\ \texttt{in}\ E_2
\end{aligned}
$$

Language: Type System

Example typing rule:

$$\frac{\Gamma \vdash E_1 : \texttt{Prob}\ T_1 \qquad \Gamma, x : T_1 \vdash E_2 : \texttt{Prob}\ T_2}{\Gamma \vdash \texttt{let}\ x \sim E_1\ \texttt{in}\ E_2 : \texttt{Prob}\ T_2}$$

Language: Type System

Example typing rule:

$$\frac{\Gamma \vdash E_1 : \texttt{Prob}\ T_1 \qquad \Gamma, x : T_1 \vdash E_2 : \texttt{Prob}\ T_2}{\Gamma \vdash \texttt{let}\ x \sim E_1\ \texttt{in}\ E_2 : \texttt{Prob}\ T_2}$$

Pass:

$\texttt{var}\ U\ \sim\ \texttt{uniform in}$

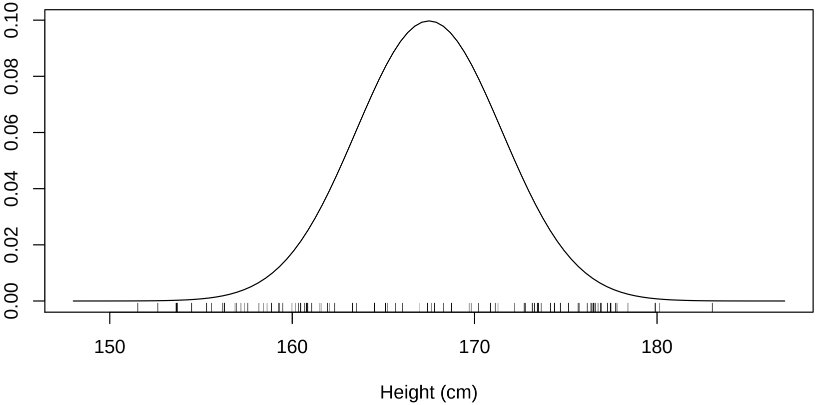
$\quad \texttt{return}\ (U \leq 0.7)$

Fail:

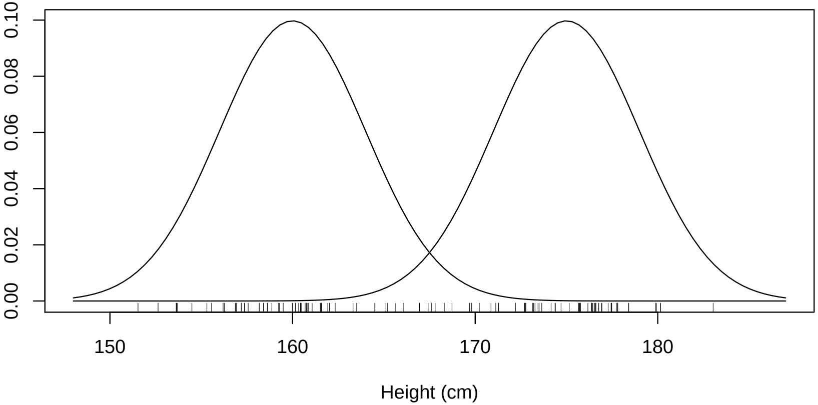
$\texttt{var}\ U\ \sim\ \texttt{uniform in}$

$\quad (U \leq 0.7)$

Gaussian Model



Mixture of Gaussians Model



Trying alternative statistical models

Formulation:

$$X_i \sim \text{Normal}(\theta, 1)$$

$$\hat{\theta} = \arg\max_{\theta} f(x \mid \theta)$$

Trying alternative statistical models

Formulation:

$$X_i \sim \text{Normal}(\theta, 1)$$
$$\hat{\theta} = \arg\max_{\theta} f(x \mid \theta)$$

Solution:

$$\hat{\theta} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

Trying alternative statistical models

Formulation:

$$X_i \sim \mathrm{Normal}(\theta, 1)$$
$$\hat{\theta} = \arg\max_{\theta} f(x \mid \theta)$$

Solution:

$$\hat{\theta} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

Formulation:

$$Z_i \sim \mathrm{Bernoulli}(0.5)$$
$$X_i \sim \mathrm{Normal}((1 - Z_i)\theta_0 + Z_i\theta_1, 1)$$
$$\hat{\theta} = \arg\max_{\theta} f(x \mid \theta)$$

Trying alternative statistical models

Formulation:

$$X_i \sim \text{Normal}(\theta, 1)$$
$$\hat{\theta} = \arg\max_{\theta} f(x \mid \theta)$$

Solution:

$$\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Formulation:

$$Z_i \sim \text{Bernoulli}(0.5)$$
$$X_i \sim \text{Normal}((1 - Z_i)\theta_0 + Z_i\theta_1, 1)$$
$$\hat{\theta} = \arg\max_{\theta} f(x \mid \theta)$$

Solution:

```
(θ̂0, θ̂1) := rand();
while (...)
   for i = 1 to n do
      γi := ϕ(xi; θ̂1,1)/(ϕ(xi; θ̂0,1)+ϕ(xi; θ̂1,1));
   θ̂0 := ∑_{i=1}^n (1-γi)*xi / ∑_{i=1}^n (1-γi);
   θ̂1 := ∑_{i=1}^n γi*xi / ∑_{i=1}^n γi;
return (θ̂0, θ̂1);
```

Interactive algorithm assistant

Bhat, Agarwal, Gray, Vuduc (2010)

Features

- enter problems
- apply schemas
- undo/redo
- combinators

Status

- can solve several textbook examples of MLE, incl. via EM
- autotuning + more sophisticated code generation is planned

```
File  Edit  View  Terminal  Help
sooraj@lucy:~/mathProg/om$ om
        Objective Caml version 3.11.1

# load gaussian;;
- : Om.Syntax.expr =
argmax{mu : R, ss : R}{
  pdf
  (let pick = normal mu ss in
   var x1 ~ pick in var x2 ~ pick in var x3 ~ pick in return (x1, x2, x3))
  (9, 28, 11)
  | 0 <= ss}
# ap ( let_simpl <&> pdf_simpl );;
- : Om.Syntax.expr =
argmax{mu : R, ss : R}{
  ss^-1.500000 * %e^((9 - mu)^2/ss * -0.500000 + (28 - mu)^2/ss *
  -0.500000 + (11 - mu)^2/ss * -0.500000) * (2 * %pi)^-1.500000
  | 0 <= ss}
# ap ( argmax_log <&> log_simpl <&> argmax_add );;
- : Om.Syntax.expr =
argmax{mu : R, ss : R}{
  -1.500000 * log ss + (9 - mu)^2/ss * -0.500000 + (28 - mu)^2/ss *
  -0.500000 + (11 - mu)^2/ss * -0.500000
  | 0 <= ss}
# ap descartes;;
- : Om.Syntax.expr =
argmax{mu : R, ss : R}{
  -1.500000 * log ss + (9 - mu)^2/ss * -0.500000 + (28 - mu)^2/ss *
  -0.500000 + (11 - mu)^2/ss * -0.500000
  | 0 <= ss && 0 = -1.500000/ss + (9 - mu)^2 * ss^-2 * 0.500000 + (28 - mu)^
    2 * ss^-2 * 0.500000 + (11 - mu)^2 * ss^-2 * 0.500000 && 0 = 1/ss * (9 -
    mu) + 1/ss * (28 - mu) + 1/ss * (11 - mu)}
# ap ( rewrite undistr <&> rewrite factors_0 <&> simpl <&> back_solve None );;
- : Om.Syntax.expr =
argmax{mu : R, ss : R}{
  -1.500000 * log ss + (9 - mu)^2/ss * -0.500000 + (28 - mu)^2/ss *
  -0.500000 + (11 - mu)^2/ss * -0.500000
  | mu = 16.000000 && ss = 72.666667}
#
```

Conclusions

- Automated bigM and convex-hull methods
- Beginnings of a formalization of probability and statistics
- Library of transformations
- Formalization of indexing provides:
    - advances on previous MP languages: *e.g.* GAMS, AMPL, OPL
    - fundamental improvements in time and space performance possible

Conclusions

- Automated bigM and convex-hull methods
- Beginnings of a formalization of probability and statistics
- Library of transformations
- Formalization of indexing provides:
  - advances on previous MP languages: *e.g.* GAMS, AMPL, OPL
  - fundamental improvements in time and space performance possible
  - challenges remain:
    e.g. conversion of
    $$\bigvee_{i:\sigma} \bigwedge_{i':\sigma'} e$$
    to indexed CNF
    $$\bigwedge_{f:(i:\sigma \to \sigma')} \bigvee_{i:\sigma} \{f\,(i)\,/i'\}\,e$$
    not supported in current theory.