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* *(Georgia Tech)*
  *Rich Vuduc* *(Georgia Tech, 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)

\[
\begin{align*}
\text{max } & \quad x_1 - x_2 \\
x_1 & \geq 1.0 \\
x_2 & \geq 1.0 \\
x_1 + x_2 & \leq 5.0
\end{align*}
\]
Cannot represent multiple polyhedra.
Declaring discrete choice – with disjunction


\[
\begin{align*}
\begin{bmatrix}
  x_1 & \geq & 1 \\
  x_2 & \geq & 1 \\
  x_1 + x_2 & \leq & 5 \\
\end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
\begin{bmatrix}
  5 & \leq & x_1 & \leq & 8 \\
  4 & \leq & x_2 & \leq & 7 \\
\end{bmatrix}
\end{align*}
\]
Declaring discrete choice – with disjunction


Language of DP extends LP with disjunction
Declaring discrete choice – with disjunction


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 $y \in \{0, 1\}$

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

- 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
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)  →  Convert to MILP (accepted by solvers)
Multiple Transformation Techniques Available
Choice affects computational efficiency

- best big-M reformulation
- convex-hull reformulation
System overview

disjunctive constraint
\[ [A^1x \leq b^1] \lor [A^2x \leq b^2] \]

\[ y = 0 \Rightarrow A^1x \leq b^1 \]
\[ y = 1 \Rightarrow A^2x \leq b^2 \]

Boolean propositions

Pure integer inequalities
Convex-hull reformulation of

\[ [A^1 x \leq b^1] \lor [A^2 x \leq b^2] \]

is

\[
\begin{align*}
A^1 \bar{x}^1 &\leq b^1 y_1 & y_1 + y_2 &= 1 \\
A^2 \bar{x}^2 &\leq b^2 y_2 & x &= \bar{x}^1 + \bar{x}^2
\end{align*}
\]
Convex-hull reformulation of

\[ [A^1 x \leq b^1] \lor [A^2 x \leq b^2] \]

is

\[
\begin{align*}
A^1 \bar{x}^1 & \leq b^1 y_1 & y_1 + y_2 = 1 \\
A^2 \bar{x}^2 & \leq b^2 y_2 & x = \bar{x}^1 + \bar{x}^2
\end{align*}
\]

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)

\[ \rho ::= [r_L, r_U] \mid [r_L, \infty) \mid (-\infty, r_U] \mid \text{real} \]
\[ \mid \langle r_L, r_U \rangle \mid \langle r_L, \infty \rangle \mid (-\infty, r_U) \mid \text{int} \]
\[ \mid \{\text{true}\} \mid \{\text{false}\} \mid \text{bool} \]
\[ e ::= x \mid r \mid \text{true} \mid \text{false} \mid \text{not } e \mid e_1 \text{ or } e_2 \mid e_1 \text{ and } e_2 \]
\[ \mid -e \mid e_1 + e_2 \mid e_1 - e_2 \mid e_1 \times e_2 \]
\[ c ::= T \mid F \mid \text{isTrue } e \mid e_1 = e_2 \mid e_1 \leq e_2 \mid c_1 \lor c_2 \mid c_1 \land c_2 \mid \exists x: \rho \cdot c \]
\[ \rho ::= \text{max}_{x_1: \rho_1, \ldots, x_m: \rho_m} \{ e \mid c \} \]
\[ \Upsilon ::= \bullet \mid \Upsilon, x: \rho \]
Convex-hull transformation \[ \gamma \vdash c \xrightarrow{\text{CVX}} c' \]

Agarwal, Bhat, Gray, Grossmann (2010)

\[
\begin{align*}
\{ \gamma \vdash c_j \xrightarrow{\text{PROP}} c'_j \} & \quad \text{if } j \in \{A,B\} \\
\{ \gamma' \vdash c'_j \xrightarrow{\mathcal{X}} x_1, \ldots, x_m \xrightarrow{\mathcal{X}} c''_j \} & \quad \text{if } j \in \{A,B\} \\
\{ y_j \odot \{ \tilde{x}^j / \tilde{x} \} & \quad \text{if } j \in \{A,B\} \\
\gamma \vdash c_A \lor c_B & \xrightarrow{\text{CVX}} \left( \exists \tilde{x}^A : \tilde{\rho} \cdot \exists \tilde{x}^B : \tilde{\rho} \cdot \exists y^A : \langle 0, 1 \rangle \cdot \exists y^B : \langle 0, 1 \rangle. \right. \\
& \quad \left. (\tilde{x} = \tilde{x}^A + \tilde{x}^B) \land (y^A + y^B = 1) \land (c''_A \cap c''_B) \right)
\end{align*}
\]

- Compile disjuncts and context
- Add bounding constraints
- Substitute disaggregated variables in each disjunct
- Multiply constant terms by respective \(y\)
Example: single disjunctive constraint

Input

\[
\begin{align*}
\text{var } x & : (10.0, 100.0) \\
\text{var } w & : (2.0, 50.0) \\
\text{min } x + w \text{ subject to} \\
(x & \leq w) \text{ disj } (x \geq w + 4.0)
\end{align*}
\]

Output

\[
\begin{align*}
\text{var } x & : (10.0, 100.0) \\
\text{var } w & : (2.0, 50.0) \\
\text{min } x + w \text{ subject to} \\
& \exists y_1 : [0, 1] \\
& \exists y_2 : [0, 1] \\
& \exists x_1 : (0.0, 100.0) \\
& \exists x_2 : (0.0, 100.0) \\
& \exists w_1 : (0.0, 50.0) \\
& \exists w_2 : (0.0, 50.0) \\
& w = w_1 + w_2, \\
& x = x_1 + x_2, \\
& y_1 + y_2 = 1, \\
& 10.0 \cdot y_1 \leq x_1, \\
& x_1 \leq 100.0 \cdot y_1, \\
& 2.0 \cdot y_1 \leq w_1, \\
& w_1 \leq 50.0 \cdot y_1, \\
& x_1 \leq w_1, \\
& 10.0 \cdot y_2 \leq x_2, \\
& x_2 \leq 100.0 \cdot y_2, \\
& 2.0 \cdot y_2 \leq w_2, \\
& w_2 \leq 50.0 \cdot y_2, \\
& x_2 \geq w_2 + 4.0 \cdot y_2
\end{align*}
\]
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
```

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

\[ \text{switched flow process} \]

\[ \dot{m}^\alpha \quad \text{on} \quad \alpha \quad \text{off} \quad \dot{m}^\beta \quad \text{hi} \quad \text{low} \]

\[ M^{\max} \quad M^{\min} \quad F^{\text{out}} \]
Pump $\alpha$ has three kinds of mode transition dynamics:

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

$$\begin{bmatrix}
isTrue \ Z^\alpha(\text{on}, \text{off}, i) \\
\hat{c}^\alpha(i) = 0.0 \\
\hat{r}^\alpha(i) = -R^e(i)
\end{bmatrix} \lor 
\begin{bmatrix}
isTrue \ Z^\alpha(\text{off}, \text{on}, i) \\
R^e(i) \geq 2.0 \\
\hat{c}^\alpha(i) = 50.00 \\
\hat{r}^\alpha(i) = -R^e(i)
\end{bmatrix} \lor 
\begin{bmatrix}
isTrue \ YY^\alpha(i) \\
\hat{c}^\alpha(i) = 0.0 \\
\hat{r}^\alpha(i) = 0.0
\end{bmatrix}$$

Booleans and disjunction enable the natural modeling of such logical relations between constraints.
Switched flow process, example constraint

Pump $\alpha$ has three kinds of mode transition dynamics:

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

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

...which interact with each other:

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

Booleans and disjunction enable the natural modeling of such logical relations between constraints.
Switched flow process, comparison to ILOG Concert

<table>
<thead>
<tr>
<th>Method</th>
<th>#vars (#binary)</th>
<th>#constr. (#IC)</th>
<th>time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>flow-Concert</td>
<td>1061 (874)</td>
<td>1080 (718)</td>
<td>36.85</td>
</tr>
<tr>
<td>flow-IC</td>
<td>477 (291)</td>
<td>1001 (438)</td>
<td>11.60</td>
</tr>
<tr>
<td>flow-BM</td>
<td>477 (291)</td>
<td>1198</td>
<td>3.37</td>
</tr>
<tr>
<td>flow-CH</td>
<td>1194 (631)</td>
<td>2747</td>
<td>1.09</td>
</tr>
</tbody>
</table>
Switched flow process, comparison to ILOG Concert

<table>
<thead>
<tr>
<th>Method</th>
<th>#vars (#binary)</th>
<th>#constr. (#IC)</th>
<th>time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>flow-Concert</td>
<td>1061 (874)</td>
<td>1080 (718)</td>
<td>36.85</td>
</tr>
<tr>
<td>flow-IC</td>
<td>477 (291)</td>
<td>1001 (438)</td>
<td>11.60</td>
</tr>
<tr>
<td>flow-BM</td>
<td>477 (291)</td>
<td>1198</td>
<td>3.37</td>
</tr>
<tr>
<td>flow-CH</td>
<td>1194 (631)</td>
<td>2747</td>
<td>1.09</td>
</tr>
</tbody>
</table>

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

Optimal Length: 27
Strip packing, formulation

The most natural formulation uses disjunction.

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

\((x_i, y_i)\) is the position of the top-left corner of rectangle \(i\).
## Strip packing, comparison to expert

<table>
<thead>
<tr>
<th>Method</th>
<th>#vars (#binary)</th>
<th>#constr. (#IC)</th>
<th>time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>pack12-IC</td>
<td>289 (264)</td>
<td>342 (264)</td>
<td>1.83</td>
</tr>
<tr>
<td>pack12-BM</td>
<td>289 (264)</td>
<td>342</td>
<td>1.22</td>
</tr>
<tr>
<td>pack12-CH</td>
<td>1345 (264)</td>
<td>2718</td>
<td>168.38</td>
</tr>
<tr>
<td>pack12-BM-expert</td>
<td>289 (264)</td>
<td>342</td>
<td>1.82</td>
</tr>
<tr>
<td>pack12-CH-expert</td>
<td>1345 (264)</td>
<td>1662</td>
<td>149.57</td>
</tr>
<tr>
<td>pack21-IC</td>
<td>883 (840)</td>
<td>1071 (840)</td>
<td>24.44</td>
</tr>
<tr>
<td>pack21-BM</td>
<td>883 (840)</td>
<td>1071</td>
<td>55.01</td>
</tr>
<tr>
<td>pack21-CH</td>
<td>4243 (840)</td>
<td>8631</td>
<td>991.68</td>
</tr>
<tr>
<td>pack21-BM-expert</td>
<td>883 (840)</td>
<td>1071</td>
<td>29.56</td>
</tr>
<tr>
<td>pack21-CH-expert</td>
<td>4243 (840)</td>
<td>5271</td>
<td>&gt; 3600.00</td>
</tr>
</tbody>
</table>
## Strip packing, comparison to expert

<table>
<thead>
<tr>
<th>Method</th>
<th>#vars (#binary)</th>
<th>#constr. (#IC)</th>
<th>time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>pack12-IC</td>
<td>289 (264)</td>
<td>342 (264)</td>
<td>1.83</td>
</tr>
<tr>
<td>pack12-BM</td>
<td>289 (264)</td>
<td>342</td>
<td>1.22</td>
</tr>
<tr>
<td>pack12-CH</td>
<td>1345 (264)</td>
<td>2718</td>
<td>168.38</td>
</tr>
<tr>
<td>pack12-BM-expert</td>
<td>289 (264)</td>
<td>342</td>
<td>1.82</td>
</tr>
<tr>
<td>pack12-CH-expert</td>
<td>1345 (264)</td>
<td>1662</td>
<td>149.57</td>
</tr>
<tr>
<td>pack21-IC</td>
<td>883 (840)</td>
<td>1071 (840)</td>
<td>24.44</td>
</tr>
<tr>
<td>pack21-BM</td>
<td>883 (840)</td>
<td>1071</td>
<td>55.01</td>
</tr>
<tr>
<td>pack21-CH</td>
<td>4243 (840)</td>
<td>8631</td>
<td>991.68</td>
</tr>
<tr>
<td>pack21-BM-expert</td>
<td>883 (840)</td>
<td>1071</td>
<td>29.56</td>
</tr>
<tr>
<td>pack21-CH-expert</td>
<td>4243 (840)</td>
<td>5271</td>
<td>&gt; 3600.00</td>
</tr>
</tbody>
</table>

- Our mechanizations perform just as well as expert encodings.
We solved a problem with 150,000 equations and 25,000 variables.
How were so many equations and variables declared?
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:
  \[ \forall i \in [1 \ldots n] \cdot x_{i+1} = x_i + y_i \]
- Indexing is a meta-programming feature
- Index variables \( i \) distinct from mathematical variables \( x \)
Complex Index Sets Arise in Real Problems

- Job shop scheduling:

\[ \forall j \in J \cdot \forall s \in S_j \cdot \forall j' \in \text{Pre}_{j,s} \cdot t_{j',s} \leq t_{j,s} \]

- 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

\[ \varepsilon ::= i \mid k \]
\[ \mid (\varepsilon_1, \ldots, \varepsilon_m) \mid \varepsilon.k \]
\[ \mid -\varepsilon \mid \varepsilon_1 + \varepsilon_2 \mid \varepsilon_1 - \varepsilon_2 \mid \varepsilon_1 \ast \varepsilon_2 \]
\[ \mid \text{case } \varepsilon \text{ of } \{ k_j \Rightarrow \varepsilon_j \}_{j=1}^{m} \]

• Index Sets (Types)

\[ \sigma ::= [\varepsilon_L..\varepsilon_U] \mid i_1:\sigma_1 \times \cdots \times i_m:\sigma_m \]
\[ \mid \text{case } \varepsilon \text{ of } \{ k_j \Rightarrow \sigma_j \}_{j=1}^{m} \]
\[ \mid \lambda i.\sigma \mid \sigma \varepsilon \]
\[ \mid \sigma :: \kappa \]

• Kinds

\[ \kappa ::= \text{IndexSet} \mid i:\sigma \Rightarrow \kappa \]
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:

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

- Other software expand this to:

\[
\begin{align*}
x_2 &= x_1 + y_1 \\
x_3 &= x_2 + y_2 \\
x_4 &= x_3 + y_3 \\
x_5 &= x_4 + y_4 \\
\vdots & \quad \vdots & \quad \vdots
\end{align*}
\]
Memory Reduction

- Load this program:

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

- Other software expand this to:

\[ x_2 = x_1 + y_1 \]
\[ x_3 = x_2 + y_2 \]
\[ x_4 = x_3 + y_3 \]
\[ x_5 = x_4 + y_4 \]
\[ \vdots \quad \vdots \quad \vdots \]

- We retain indexing structure:

Memory requirements reduced from \( O(n) \) to \( O(1) \).
Computational Improvements

- **Input to our software:**
  \[
  \bigwedge_{i:[1..10]} w \geq x_i + 4.0
  \]

- **Our software’s output:**
  \[
  \bigwedge_{i:[1..10]} \begin{bmatrix}
  10.0 \times y_i \leq w'_i \\
  w'_i \leq 90.0 \times y_i \\
  w'_i \geq x'_{i,i} + 4.0 \times y_i
  \end{bmatrix}
  \bigwedge_{d:[1..10]} \begin{bmatrix}
  5.0 \times y_i \leq x'_{i,d} \\
  x'_{i,d} \leq 75.0 \times y_i
  \end{bmatrix}
  \]
Computational Improvements

- **Input to our software:**

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

- **Our software’s output:**

\[
\begin{align*}
\bigwedge_{i: [1..10]} & \left[ \begin{array}{c}
10.0 \times y_i \leq w'_i \\
w'_i \leq 90.0 \times y_i \\
w'_i \geq x'_{i,i} + 4.0 \times y_i \\
5.0 \times y_i \leq x'_{i,d} \\
x'_{i,d} \leq 75.0 \times y_i \\
\end{array} \right] \\
\bigwedge_{d: [1..10]} & \left[ x'_{i,d} \leq 75.0 \times y_i \right]
\end{align*}
\]

Reformulation time reduced from \( O(n) \) to \( O(1) \).
Usual Way: Syntax $\rightarrow$ Type System $\rightarrow$ Semantics
- Existence is prior to meaning.

Alternative Way: Syntax $\rightarrow$ Semantics $\rightarrow$ Type System
- Meaning is prior to existence.
Indexing Language: Type System and Semantics
Agarwal (2006)

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

  \[ X : \Omega \rightarrow \mathbb{R} \]

  that assigns a real number \( X(\omega) \) to each outcome \( \omega \).
What Are Random Variables?

- Wasserman (2004) says:

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

  However:

- Treated as real: \( \mathbb{P}(X \geq 5) \)

- Not random:
  We write
  \[ X \sim \text{Bernoulli}(p) \]
  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 $\text{Bernoulli}(0.5)$, 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

  Probability distributions are a monad.

- Kozen (1981)
  Formalized semantics.

- Ramsey and Pfeffer (2002)
  Efficient expectations, but discrete distributions only.

  Continuous distributions also, but support only sampling.

- Erwig and Kollmansberger (2006)
  Provide Haskell library, but discrete distributions only, computational efficiency not optimized.
Previous Work

  Probability distributions are a monad.
- Kozen (1981)
  Formalized semantics.
- Ramsey and Pfeffer (2002)
  Efficient expectations, but discrete distributions only.
  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)

\[
T ::= \text{Bool} \mid \text{Int} \mid \text{Real} \mid T_1 \times T_2 \mid \text{Prob } T \\
E ::= x \mid \text{true} \mid \text{false} \\
      \mid r \mid E_1 + E_2 \mid E_1 \times E_2 \\
      \mid (E_1, E_2) \mid \text{fst } E \mid \text{snd } E \\
      \mid \text{if } E_1 \text{ then } E_2 \text{ else } E_3 \mid E_1 = E_2 \mid E_1 \leq E_2 \\
      \mid \text{uniform} \mid \text{return } E \mid \text{let } x \sim E_1 \text{ in } E_2
\]
Example typing rule:

\[
\Gamma \vdash E_1 : \text{Prob } T_1 \quad \Gamma, x : T_1 \vdash E_2 : \text{Prob } T_2
\]

\[
\Gamma \vdash \text{let } x \sim E_1 \text{ in } E_2 : \text{Prob } T_2
\]
Language: Type System

Example typing rule:

\[
\Gamma \vdash E_1 : \text{Prob } T_1 \quad \Gamma, x : T_1 \vdash E_2 : \text{Prob } T_2 \\
\Gamma \vdash \text{let } x \sim E_1 \text{ in } E_2 : \text{Prob } T_2
\]

**Pass:**

- \( \text{var } U \sim \text{uniform in } \)
- \( \text{return } (U \leq 0.7) \)

**Fail:**

- \( \text{var } U \sim \text{uniform in } \)
- \( (U \leq 0.7) \)
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:

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

Solution:

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

Formulation:

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

\[
(\hat{\theta}_0, \hat{\theta}_1) := \text{rand}();
\]

while (...)

\[
\text{for } i = 1 \text{ to } n \text{ do}
\]

\[
\gamma_i := \phi(x_i; \hat{\theta}_1, 1) / (\phi(x_i; \hat{\theta}_0, 1) + \phi(x_i; \hat{\theta}_1, 1));
\]

\[
\hat{\theta}_0 := \sum_{i=1}^{n} (1-\gamma_i) * x_i / \sum_{i=1}^{n} (1-\gamma_i);
\]

\[
\hat{\theta}_1 := \sum_{i=1}^{n} \gamma_i * x_i / \sum_{i=1}^{n} \gamma_i;
\]

return \((\hat{\theta}_0, \hat{\theta}_1)\);
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
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
    \[
    \bigwedge_{f:(i:\sigma \rightarrow \sigma')} \bigvee_{i:\sigma} \{ f(i) / i' \} e
    \]
    not supported in current theory.