XML-based standard

Optimization Services (OS) Framework
THE Optimization Internet

Model/Data

Parse to OSiL

AMPL

OSmL

Agent

OSP/OSxL

Solver

CGI

location

Web address

Web page

OS Server

OS Server

OS Server

Hospital/ App Service

HTML Checker

Data in HTML Form

Raw Data

Analyzer

:objective

Max \( f(x) \)

:constraints

s.t. \( lb_1 \leq g_1(x) \leq ub_2 \)

\( lb_2 \leq g_2(x) \leq ub_2 \)

:variables

\( f(x) \) can be \( \sin(x(1)) + x(x(2)) \)

\( g_1(x) \) can be \( \text{if}(x(1) > 0) \text{then} x(2) \text{else} \text{cost}(x(2)) \)

\( g_2(x) \) can be a metric from a finite element simulation

(non-closed form black box function evaluator)

OSiL – Optimization Services Instance Language

Modeler

html form

browser

socket

http/html

Data in HTML Form

Google

Robert Fourer, Jun Ma, Kipp Martin
Copyright 2004
OUTLINE

1. Introduction and motivation
2. Model versus Instance
3. Why XML?
4. The OSiL Schema
5. Conclusion
There is a proliferation of modeling languages and solvers

<table>
<thead>
<tr>
<th>AIMMS</th>
<th>CLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMPL</td>
<td>CPLEX</td>
</tr>
<tr>
<td>GAMS</td>
<td>GLPK</td>
</tr>
<tr>
<td>LINGO</td>
<td>LINDO</td>
</tr>
<tr>
<td>OSmL</td>
<td>MINOS</td>
</tr>
<tr>
<td>MPL</td>
<td>MOSEK</td>
</tr>
<tr>
<td>OPL</td>
<td>Xpress-MP</td>
</tr>
</tbody>
</table>
Consequence:
a lot of drivers are needed for every modeling language to talk to every solver
It would be nice to have an instance representation language.
A MODEL

set PROD;  # products
set DEP;   # processing departments

param hours {DEP};   # time available in each department
param rate {DEP,PROD}; # hours used in each dept per product unit made
param profit {PROD};  # profit per unit of each product made

var Make {PROD} >= 0;   # number of units of each product to be made

maximize TotalProfit:
sum {j in PROD} profit[j] * Make[j];

subject to HoursAvailable {i in DEP}:
sum {j in PROD} rate[i,j] * Make[j] <= hours[i];
DATA

param: PROD: profit :=
    std  10
    del  9;

param: DEP: hours :=
    cutanddye  630
    sewing  600
    finishing  708
    inspectandpack  135;

param: rate: std  del :=
    cutanddye  0.7  1.0
    sewing  0.5  0.8333
    finishing  1.0  0.6667
    inspectandpack  0.1  0.25;
MODEL + DATA = INSTANCE

maximize TotalProfit:
10*Make[‘std’] + 9*Make[‘del’];

subject to HoursAvailable[‘cutanddye’]:
0.7*Make[‘std’] + Make[‘del’] <= 630;

subject to HoursAvailable[‘sewing’]:
0.5*Make[‘std’] + 0.8333*Make[‘del’] <= 600;

subject to HoursAvailable[‘finishing’]:
Make[‘std’] + 0.6667*Make[‘del’] <= 708;

subject to HoursAvailable[‘inspectandpack’]:
0.1*Make[‘std’] + 0.25*Make[‘del’] <= 135;
## Why not MPS?

<table>
<thead>
<tr>
<th>NAME</th>
<th>PRODMIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROWS</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>TPROFIT</td>
</tr>
<tr>
<td>L</td>
<td>HRSCUT</td>
</tr>
<tr>
<td>L</td>
<td>HRSSEW</td>
</tr>
<tr>
<td>L</td>
<td>HRSFIN</td>
</tr>
<tr>
<td>L</td>
<td>HRSINS</td>
</tr>
<tr>
<td>COLUMNS</td>
<td></td>
</tr>
<tr>
<td>MAKESTD</td>
<td>TPROFIT</td>
</tr>
<tr>
<td>MAKESTD</td>
<td>HRSCUT</td>
</tr>
<tr>
<td>MAKESTD</td>
<td>HRSFIN</td>
</tr>
<tr>
<td>MAKEDEL</td>
<td>TPROFIT</td>
</tr>
<tr>
<td>MAKEDEL</td>
<td>HRSCUT</td>
</tr>
<tr>
<td>MAKEDEL</td>
<td>HRSFIN</td>
</tr>
<tr>
<td>RHS</td>
<td></td>
</tr>
<tr>
<td>RHS1</td>
<td>HRSCUT</td>
</tr>
<tr>
<td>RHS1</td>
<td>HRSSEW</td>
</tr>
<tr>
<td>RHS1</td>
<td>HRSFIN</td>
</tr>
<tr>
<td>RHS1</td>
<td>HRSINS</td>
</tr>
<tr>
<td>ENDATA</td>
<td></td>
</tr>
</tbody>
</table>
The Case for XML

1. Validation against a schema provides for error checking

2. Validation against a schema promotes stability of a standard

3. The schema can restrict data values to appropriate types, e.g. row names to string, indices to integer, coefficients to double

4. The schema can define keys to insure, for example, no row or column name is used more than once.

5. The schema can be extended to include new constraint types or solver directives

6. There is a lot of open source software to make parsing easy.
XML and Optimization Systems

1. When instances are stored in XML format, optimization technology solutions are more readily integrated into broader IT infrastructures

2. XML is used for Web Services – important for distributed computing

3. The XML format lends itself well to compression – more on this later

4. The XML format can be combined with other technologies, e.g. XSLT to present results in human readable formats
XML Concepts

XML (Extensible Markup Language) – an XML file contains both data and Markup (Elements (tags) and Attributes)

The tags are organized in a tree like structure. The closing tag of a child element preceding the closing tag of its parent.

<rows>
  <row rowName="cutanddye" rowUB="630"/>
  <row rowName="sewing" rowUB="600"/>
  <row rowName="finishing" rowUB="708"/>
  <row rowName="inspectandpack" rowUB="135"/>
</rows>
XML Schema Concepts – a Row Class

<xs:element name="rows">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="row" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:attribute name="rowName" type="xs:string" use="optional"/>
          <xs:attribute name="rowUB" type="xs:double" use="optional"/>
          <xs:attribute name="rowLB" type="xs:double" use="optional"/>
          <xs:attribute name="mult" type="xs:int" use="optional"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
XML Concepts – a Row Object

<rows>
  <row rowName="cutanddye" rowUB="630"/>
  <row rowName="sewing" rowUB="600"/>
  <row rowName="finishing" rowUB="708"/>
  <row rowName="inspectandpack" rowUB="135"/>
</rows>
1. General Architecture

- OSiL Schema

```
OSiL

programDescription

programData

This is the only allowed root element.
```

Robert Fourer, Jun Ma, Kipp Martin
Copyright 2004
OSiL Schema

2. Information about the instance
   No Information on Solver (OSoL)
   *No Meta-Information (OSaL)
   No Result Information (OSrL)
2. The instance data

OSiL Schema

Originates from LPFML
OSiL Schema

2. The instance data (rows and columns)

<rows>
  <row rowName="HoursAvailable['cutanddye']" rowUB="630"/>
  <row rowName="HoursAvailable['sewing']" rowUB="600"/>
  <row rowName="HoursAvailable['finishing']" rowUB="708"/>
  <row rowName="HoursAvailable['inspectandpack']" rowUB="135"/>
</rows>

<columns>
  <col objVal="10" colName="Make['std']" colType="C" colLB="0.0"/>
  <col objVal="9" colName="Make['del']" colType="C" colLB="0.0"/>
</columns>
2. The instance data (the A matrix)
2. The instance data (the A matrix)

```
<sparseMatrix>
  <pntANonz>
    <el>2</el><el>4</el>
  </pntANonz>
  <rowIdx>
    <el>1</el><el>2</el><el>0</el><el>1</el>
  </rowIdx>
  <nonz>
    <el>1</el><el>2</el><el>-3</el><el>4</el>
  </nonz>
</sparseMatrix>
```

\[
\begin{bmatrix}
0 & -3 \\
1 & 4 \\
2 & 0 \\
\end{bmatrix}
\]
3. Extensions
Conclusion and Extension

1. The libraries are to be open source. Check http://www.optimizationservices.org

2. Libraries to be licensed under a non-copyleft license

3. Extensions include Networks, SDP, Quadratic, Complementarity, Stochastic Programming, Constraint/Logic/Combinatorial, General Nonlinear