OSiL: An Open Standard for Expressing and Using Optimization Problem Instances

Robert Fourer, Jun Ma
Industrial Engineering & Management Sciences
Northwestern University
[4er,maj]@iems.northwestern.edu

Kipp Martin
Graduate School of Business
University of Chicago
kmartin@gsb.uchicago.edu

EURO XXI — 21st European Conference on Operational Research
Reykjavik, Iceland, July 2-5, 2006
Session WA27.1, COIN-OR II
XML-Based Standard Formats

Motivation
- for any standard format
- for an XML-based format

“OSxL” standards
- OSiL: problem instances
- OSoL: solver options
- OSrL: results

... and a host of others
(see www.optimizati0nservices.org)

Components of OSiL
- XML schema for text file format, and
- Corresponding in-memory data structures
- Libraries for reading and writing the above
Standards

XML Means “Tagged” Text Files . . .

Example: html for a popular home page

```
<html><head><meta http-equiv="content-type" content="text/html; charset=UTF-8"><title>Google</title><style><!--body,td,a,p,.h{font-family:arial,sans-serif;}
 .h{font-size: 20px;}
 .q{text-decoration:none; color:#0000cc;}
//-->
</style></head><body bgcolor=#ffffff text=#000000
vlink=#551a8b alink=#ff0000 onLoad=sf()><center><table border=0
cellspacing=0 cellpadding=0><tr><td><img src="/images/logo.gif" width=276 height=110
alt="Google"></td></tr></table><br>
.......<font size=-2>&copy;2003 Google - Searching 3,307,998,701 web
pages</font></center></body></html>
```

. . . a collection of XML tags is designed for a special purpose
. . . by use of a schema written itself in XML
Advantage of any standard

**MN drivers**
without a standard

**M + N drivers**
with a standard
Advantages of an XML Standard

Specifying it
- Unambiguous definition via a schema
- Provision for keys and data typing
- Well-defined expansion to new name spaces

Working with it
- Parsing and validation via standard utilities
- Amenability to compression and encryption
- Transformation and display via XSLT style sheets
- Compatibility with web services
What about “MPS Form”?

Weaknesses

- Standard only for LP and MIP, not for nonlinear, network, complementarity, logical, . . .
- Standard not uniform (especially for SP extension)
- Verbose ASCII form, with much repetition of names
- Limited precision for some numerical values

Used for

- Collections of (mostly anonymous) test problems
- Bug reports to solver vendors

Not used for

- Communication between modeling systems and solvers
**Standards**

**Text from the OSiL Schema**

```
<xs:complexType name="Variables">
  <xs:sequence>
    <xs:element name="var" type="Variable" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="number" type="xs:positiveInteger" use="required"/>
</xs:complexType>
```

```
<xs:complexType name="Variable">
  <xs:attribute name="name" type="xs:string" use="optional"/>
  <xs:attribute name="init" type="xs:string" use="optional"/>
  <xs:attribute name="type" use="optional" default="C">
    <xs:simpleType>
      <xs:restriction base="xs:string">  
        <xs:enumeration value="C"/>  
        <xs:enumeration value="B"/>  
        <xs:enumeration value="I"/>  
        <xs:enumeration value="S"/>  
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="lb" type="xs:double" use="optional" default="0"/>
  <xs:attribute name="ub" type="xs:double" use="optional" default="INF"/>
</xs:complexType>
```
Standards

Diagram of the OSiL Schema
Details of OSiL’s instanceData Element

- Variables
  - Number
  - Attributes
- Objectives
  - Objective
    - Attributes
    - Coef
- Constraints
  - Constraint
    - Attributes
Details of OSiL’s instanceData Element
Example: A Problem Instance (in AMPL)

ampl: expand _var;

Coefficients of x[0]:
  Con1  1 + nonlinear
  Con2  7 + nonlinear
  Obj   0 + nonlinear

Coefficients of x[1]:
  Con1  0 + nonlinear
  Con2  5 + nonlinear
  Obj   9 + nonlinear

ampl: expand _obj;

minimize Obj:
  (1 - x[0])^2 + 100*(x[1] - x[0]^2)^2 + 9*x[1];

ampl: expand _con;

subject to Con1:
  10*x[0]^2 + 11*x[1]^2 + 3*x[0]*x[1] + x[0] <= 10;

subject to Con2:
  log(x[0]*x[1]) + 7*x[0] + 5*x[1] >= 10;
Example in OSiL

<instanceHeader>
  <name>Modified Rosenbrock</name>
  <source>Computing Journal3:175-184, 1960</source>
  <description>Rosenbrock problem with constraints</description>
</instanceHeader>

<variables number="2">
  <var lb="0" name="x0" type="C"/>
  <var lb="0" name="x1" type="C"/>
</variables>

<objectives number="1">
  <obj maxOrMin="min" name="minCost" numberOfObjCoef="1">
    <coef idx="1">9</coef>
  </obj>
</objectives>

<constraints number="2">
  <con ub="10.0"/>
  <con lb="10.0"/>
</constraints>
Standard formats

Example in OSiL (continued)

```xml
<linearConstraintCoefficients numberOfValues="3">
  <start>
    <el>0</el>
    <el>1</el>
    <el>3</el>
  </start>
  <rowIdx>
    <el>0</el>
    <el>1</el>
    <el>1</el>
  </rowIdx>
  <value>
    <el>1.0</el>
    <el>7.0</el>
    <el>5.0</el>
  </value>
</linearConstraintCoefficients>

<quadraticCoefficients numberOfQPTerms="3">
  <qpTerm idx="0" idxOne="0" idxTwo="0" coef="10"/>
  <qpTerm idx="0" idxOne="1" idxTwo="1" coef="11"/>
  <qpTerm idx="0" idxOne="0" idxTwo="1" coef="3"/>
</quadraticCoefficients>
```
Standard formats

Example in OSiL (continued)

```
<nl idx="-1">
  <plus>
    <power>
      <minus>
        <number type="real" value="1.0" />
        <variable coef="1.0" idx="1" />
      </minus>
      <number type="real" value="2.0" />
    </power>
    <times>
      <power>
        <minus>
          <variable coef="1.0" idx="0" />
          <power>
            <variable coef="1.0" idx="1" />
            <number type="real" value="2.0" />
          </power>
        </minus>
        <number type="real" value="2.0" />
      </power>
      <number type="real" value="100" />
    </times>
  </plus>
</nl>
```
Standard formats

Example in OSiL (continued)

```xml
<nl idx="1">
  <ln>
    <ln>
      <times>
        <variable idx="0"/>
        <variable idx="1"/>
      </times>
    </ln>
  </ln>
</nl>
```
Compression

Specific to OSiL
- Collapse sequences of row/column numbers
- Collapse repeated element values
- Encode portions using base-64 datatype

General for XML
- Compression schemes designed for XML files

Comparisons
- XML base-64 < MPS
- XML with multiple values collapsed < 2 × MPS
- Compressed XML < Compressed MPS
Other Features in OSiL . . .

In current specification
- Real-time data
- Functions defined by the user

In process of design
- Stochastic programming / optimization under uncertainty
  * see Gus Gassmann’s talk, WB44.4: An XML-Based Schema for Stochastic Programming
- Logical / combinatorial constraints
- Semidefinite / cone programming

Associated languages
- OSoL for communicating options to solvers
- OSrL for communicating results from solvers

. . . broader family of “optimization services” languages
(see www.optimizationservices.org)
In-Memory Data Structures

**OSInstance object class**
- Parallels the OSiL schema
- complexType in schema $\leftrightarrow$ class in OSInstance
- attributes / children of an element $\leftrightarrow$ members of a class
- choices / sequences in the schema arrays $\leftrightarrow$ array members

**OS expression tree**
- Parallels the *nonlinear* part of the OSiL schema
- Designed to avoid lengthy “switch” statements

**Advantages**
- One standard instead of two
- Complements COIN-OR’s OSI
Libraries (APIs, Interfaces)

*Use by client*
- OSInstance set() methods generate instance in memory
- OSiLWriter writes instance to a file in OSiL format
- Using SOAP over HTTP, instance is sent to a solver

*Use by solver*
- OSiLReader in solver interface reads instance from OSiL format back to memory
- OSInstance get() methods extract instance data as needed by solver
- Solver works on the problem
- Results are sent back similarly, using OSrL

... OSiL can be skipped when instance is passed in memory
For More Information
