Optimization Via the Internet: NEOS 5 and Beyond

Robert Fourer
Industrial Engineering & Management Sciences
Northwestern University
Evanston, Illinois 60208-3119, U.S.A.
4er@iems.northwestern.edu — www.iems.northwestern.edu/~4er/

and many others . . .

19th International Symposium on Mathematical Programming
Rio de Janeiro, July 30 – August 4, 2006
Session WE1-R15, Software Services and Distribution
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

Jorge J. Moré, Todd S. Munson, Jason Sarich
Mathematics and Computer Science Division
Argonne National Laboratory
[more,tmunson,sarich]@mcs.anl.gov

Dominique Orban
Mathematics and Industrial Engineering Department
École Polytechnique de Montréal
dominique.orban@polymtl.ca
NEOS  www-neos.mcs.anl.gov/neos/

A general-purpose optimization server

- Over 45 solvers in all
  - Linear, linear network, linear integer
  - Nonlinear, nonlinear integer, nondifferentiable & global
  - Stochastic, semidefinite, semi-infinite, complementarity
- Commercial as well as experimental solvers
- Central scheduler with distributed solver sites

A research project

- Currently free of charge
- Supported through the Optimization Technology Center of Northwestern University and Argonne National Laboratory
NEOS Design

Flexible architecture
- Central controller and scheduler machine
- Distributed solver sites

Numerous formats
- Low-level formats: MPS, SIF, SDPA
- Programming languages: C/ADOL-C, Fortran/ADIFOR
- High-level modeling languages: AMPL, GAMS

Varied submission options
- E-mail
- Web form
- Direct call via XML-RPC
  * from AMPL or GAMS client
  * from user’s client program using NEOS’s API

... server processes submissions of new solvers, too
NEOS Frequently Asked Questions

Who uses it?
- Where are its users from?
- How much is it used?

What kinds of solvers does it offer?
- Who supplies them?
- Which are most heavily used?
- Where are they hosted?

How is it supported?
- Who answers user questions?
Who Uses NEOS? *(*a sample*)

- We are using NEOS services for duty-scheduling for ground handling activities in a regional airport environment.
- We used NEOS to solve nonlinear optimization problems associated with models of physical properties in chemistry.
- Our company is working with various projects concerning R&D of internal combustion engines for cars and brakes for heavy vehicles.
- We are working on bi-dimensional modeling of earth's conductivity distribution.
- I am dealing with ultimate limit-state analyses of large dams by means of a non-standard approach (“direct method”); this requires solving problems of linear and non-linear programming. The NEOS server is an extraordinary tool to perform parametric tests on small models, in order to choose the best suited solver.
- I have used NEOS with LOQO solver to optimize an interpolator. . . . My domain is digital receivers where the receiver clock is not changed to match the transmitter clock.
Who Uses NEOS? (more)

- I have been able to build and solve a prototype combinatorial auction MIP model using AMPL and NEOS in a fraction of the time it would have required me to do this had I needed to requisition a solver and install it locally.

- Our idea is trying to design antennas by using the computer. . . . We have tried various solvers on NEOS to see if this is possible at all.

- I am using the LOQO solver and code written in AMPL to perform numerical optimization of a spinor Bose-Einstein condensate.

- We are using the NEOS Server for solving linear and nonlinear complementarity problems in engineering mechanics and in robotics.

- I have been working on a system for protein structure prediction. . . . I had need to incorporate a nonlinear solver to handle packing of sidechain atoms in the protein.
Who Uses NEOS? (academic)

- I am regularly suggesting my students to use NEOS as soon as their projects in AMPL cannot be solved with the student edition. **So they debug their AMPL models locally . . . and then they run their real-life projects thanks to NEOS.**

- I didn’t even know what nonlinear programming was and after I discovered what it was, it became clear how enormous a task it would be to solve the problems assigned to me. . . . I had extremely complicated objective functions, both convex and nonconvex, an armload of variables, and an armload of convex, nonconvex, equality and inequality constraints, but when I sent off the information via the web submission form, within seconds I received extremely accurate and consistent results. **The results were used for verifying a certain theory in my professor’s research** and so accuracy was extremely important.

- NEOS has been a very valuable tool in the two graduate optimization courses that I teach. **NEOS allows students to see a broader variety of solvers than we have available . . .**

  ... more at [www-neos.mcs.anl.gov/neos/stories.html](http://www-neos.mcs.anl.gov/neos/stories.html)
**NEOS Users**

**Where are They From?**

*June 2005 through May 2006: Identifiable domain and >= 20 submissions*

<table>
<thead>
<tr>
<th>Domain</th>
<th>Submissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>(com)</td>
<td>12575</td>
</tr>
<tr>
<td>(edu)</td>
<td>69648</td>
</tr>
<tr>
<td>(gov)</td>
<td>34547</td>
</tr>
<tr>
<td>(net)</td>
<td>21204</td>
</tr>
<tr>
<td>(mil)</td>
<td>277</td>
</tr>
<tr>
<td>(org)</td>
<td>374</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Country</th>
<th>Submissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Argentina (ar)</td>
<td>26</td>
</tr>
<tr>
<td>Australia (au)</td>
<td>434</td>
</tr>
<tr>
<td>Austria (at)</td>
<td>171</td>
</tr>
<tr>
<td>Belgium (be)</td>
<td>5973</td>
</tr>
<tr>
<td>Brazil (br)</td>
<td>2183</td>
</tr>
<tr>
<td>Bulgaria (bg)</td>
<td>60</td>
</tr>
<tr>
<td>Canada (ca)</td>
<td>11247</td>
</tr>
<tr>
<td>Chile (cl)</td>
<td>2745</td>
</tr>
<tr>
<td>Colombia (co)</td>
<td>1201</td>
</tr>
<tr>
<td>Croatia (hr)</td>
<td>223</td>
</tr>
<tr>
<td>Cyprus (cy)</td>
<td>179</td>
</tr>
<tr>
<td>Czech Republic (cz)</td>
<td>1012</td>
</tr>
<tr>
<td>Denmark (dk)</td>
<td>94</td>
</tr>
<tr>
<td>Finland (fi)</td>
<td>46</td>
</tr>
<tr>
<td>France (fr)</td>
<td>2162</td>
</tr>
<tr>
<td>Germany ...</td>
<td></td>
</tr>
</tbody>
</table>
### NEOS Users

#### What Countries are They From?

<table>
<thead>
<tr>
<th>Country</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Argentina (ar)</td>
<td>26</td>
</tr>
<tr>
<td>Australia (au)</td>
<td>434</td>
</tr>
<tr>
<td>Austria (at)</td>
<td>171</td>
</tr>
<tr>
<td>Belgium (be)</td>
<td>5973</td>
</tr>
<tr>
<td>Brazil (br)</td>
<td>2183</td>
</tr>
<tr>
<td>Bulgaria (bg)</td>
<td>60</td>
</tr>
<tr>
<td>Canada (ca)</td>
<td>11247</td>
</tr>
<tr>
<td>Chile (cl)</td>
<td>2745</td>
</tr>
<tr>
<td>Colombia (co)</td>
<td>1201</td>
</tr>
<tr>
<td>Croatia (hr)</td>
<td>223</td>
</tr>
<tr>
<td>Cyprus (cy)</td>
<td>179</td>
</tr>
<tr>
<td>Czech Republic (cz)</td>
<td>1012</td>
</tr>
<tr>
<td>Denmark (dk)</td>
<td>94</td>
</tr>
<tr>
<td>Finland (fi)</td>
<td>46</td>
</tr>
<tr>
<td>France (fr)</td>
<td>2162</td>
</tr>
<tr>
<td>Germany (de)</td>
<td>3694</td>
</tr>
<tr>
<td>Hong Kong (hk)</td>
<td>45</td>
</tr>
<tr>
<td>Hungary (hu)</td>
<td>34</td>
</tr>
<tr>
<td>India (in)</td>
<td>704</td>
</tr>
<tr>
<td>Ireland (ie)</td>
<td>24</td>
</tr>
<tr>
<td>Italy (it)</td>
<td>2401</td>
</tr>
<tr>
<td>Japan (jp)</td>
<td>1376</td>
</tr>
<tr>
<td>Korea (kr)</td>
<td>35</td>
</tr>
<tr>
<td>Malaysia (my)</td>
<td>640</td>
</tr>
<tr>
<td>Mexico (mx)</td>
<td>163</td>
</tr>
<tr>
<td>Netherlands (nl)</td>
<td>5026</td>
</tr>
<tr>
<td>New Zealand (nz)</td>
<td>299</td>
</tr>
<tr>
<td>Norway (no)</td>
<td>156</td>
</tr>
<tr>
<td>Poland (pl)</td>
<td>195</td>
</tr>
<tr>
<td>Portugal (pt)</td>
<td>1257</td>
</tr>
<tr>
<td>Russia (ru)</td>
<td>178</td>
</tr>
<tr>
<td>Singapore (sg)</td>
<td>1324</td>
</tr>
<tr>
<td>Spain (es)</td>
<td>1265</td>
</tr>
<tr>
<td>Sweden (se)</td>
<td>3442</td>
</tr>
<tr>
<td>Switzerland (ch)</td>
<td>518</td>
</tr>
<tr>
<td>Taiwan (tw)</td>
<td>53</td>
</tr>
<tr>
<td>Turkey (tr)</td>
<td>1432</td>
</tr>
<tr>
<td>Ukraine (ua)</td>
<td>398</td>
</tr>
<tr>
<td>United Kingdom (uk)</td>
<td>2084</td>
</tr>
<tr>
<td>Uruguay (uy)</td>
<td>292</td>
</tr>
<tr>
<td>Uzbekistan (uz)</td>
<td>98</td>
</tr>
</tbody>
</table>
NEOS Users

How Much Do They Use It?

Submissions by month, 1/1999 through 5/2006

... 25 / hour over past year
... 50 / hour in peak months
What Solvers Does NEOS Offer?

For familiar problem types
- Linear programming
- Linear network optimization
- Linear integer programming
- Nonlinear programming
- Stochastic linear programming
- Complementarity problems

For emerging problem types
- Nondifferentiable optimization
- Semi-infinite optimization
- Global optimization
- Nonlinear integer programming
- Semidefinite & 2nd-order cone programming

. . . virtually every published semidefinite programming code
NEOS Solvers

Who Supplies Them?

Some commercial solver vendors
- Xpress, FortMP (mixed integer)
- CONOPT, KNITRO, MOSEK (nonlinear)

Universities and their researchers
- BonsaiG (mixed integer)
- DONLP2, FILTER, LANCELOT, LOQO, MINOS, SNOPT (nonlinear)

Open-Source Enthusiasts
- GLPK (mixed integer)

with thanks to . . .
- AMPL and GAMS developers
- Hans Mittelmann, Arizona State
NEOS Solvers

Which are Most Heavily Used?

Totals for June 2005 to May 2006
NEOS Solvers

Which are Most Heavily Used?

Totals for June 2004 to May 2005
NEOS Solvers

Where are They Hosted?

Varied workstations at

- Aachen University of Technology
- Argonne National Laboratory
- Arizona State University
- Lehigh University
- National Taiwan University
- Northwestern University (with support from Sun Microsystems)
- University of Wisconsin at Madison

... new hosts are readily added anywhere on the Internet
How is NEOS Supported?

Grants

- National Science Foundation, Operations Research Program, grant DMI-0322580
- National Science Foundation, Information Technology Research Program, grant CCR-0082807
- National Science Foundation, Challenges in Computational Science Program, grant CDA-9726385

Donations

- Processor cycles
- Many people’s time

... no user charges as yet
Who Answers Users’ Questions?

Large mailing list for support questions

- NEOS developers
- Solver developers

Support request buttons on every page
New in Version 5

Callable interface

XML-based communications

Short-job queue

New solver submission procedures
Callable Interface

Server
- at Argonne National Laboratory

Clients
- in Python, Perl, C, C++, Java, and others
- new Kestrel clients for AMPL and GAMS

... see Python example later in this talk

Methods
- for getting information from NEOS
- for submitting and retrieving jobs on NEOS
- for maintaining solvers on NEOS
Callable Interface

Getting Information

“get” methods
- `getSolverTemplate(category, solvername, inputMethod)`
- `getXML(category, name, input)`

“list” methods
- `listAllSolvers()`
- `listCategories()`
- `listSolversInCategory(category)`

Utilities
- `help()`
- `emailHelp()`
- `welcome()`
- `version()`
- `ping()`
- `printQueue()`
Submitting and Retrieving Jobs

Submission methods

- submitJob(xmlstring, user='', interface='', id=0)
- killJob(jobNumber, password, killmsg='')

Intermediate retrieval methods

- getJobStatus(jobNumber, password)
- getIntermediateResults(jobNumber, password, offset)
- getIntermediateResultsNonBlocking(jobNumber, password)

Final retrieval methods

- getFinalResults(jobNumber, password)
- getFinalResultsNonBlocking(jobNumber, password)
Callable Interface

Maintaining Solvers

Solver setup methods
- pingHost(user, hostname)
- validateSolverXML(xmlString)
- registerSolver(xmlString)

Solver management methods
- disableSolver(category, solvername, input, password)
- enableSolver(category, solvername, input, password)
- removeSolver(category, solvername, input, password)

Example methods
- registerExample(xmlstring, password)
- removeExample(category, solvername, input, password, examplename)
XML-Based Communications

<document>
<category>nco</category>
<solver>KNITRO</solver>
<inputMethod>AMPL</inputMethod>

<model><![CDATA[
...Insert Value Here...
]]></model>

<data><![CDATA[
...Insert Value Here...
]]></data>

<commands><![CDATA[
...Insert Value Here...
]]></commands>

<comments><![CDATA[
...Insert Value Here...
]]></comments>

</document>
XML

Submissions

By e-mail
- Insert actual files
- Send as text file
- Receive results via e-mail

From XML-RPC client
- Insert file names
- Submit file using `submitJob()` method
- Check status and intermediate results using appropriate methods
- Retrieve results using `getFinalResults()` method

... results include everything sent to standard output
Example: Python Client

```python
#!/usr/bin/env python
import sys
import xmlrpclib
import time

from config import Variables

if len(sys.argv) < 2 or len(sys.argv) > 3:
    sys.stderr.write("Usage: NeosClient <xmlfilename | help | queue> ")
    sys.exit(1)

neos = xmlrpclib.Server("http://%s:%d" % (Variables.NEOS_HOST, Variables.NEOS_PORT))

if sys.argv[1] == "help":
    sys.stdout.write(neos.help())

elif sys.argv[1] == "queue":
    msg = neos.printQueue()
    sys.stdout.write(msg)

else: ...
```
Example: Python Client (cont’d)

```python
xmlfile = open(sys.argv[1],"r")
xml=""
buffer=1

while buffer:
    buffer = xmlfile.read()
    xml+= buffer
xmlfile.close()

(jobNumber,password) = neos.submitJob(xml)
sys.stdout.write("jobNumber = %d " % jobNumber)
offset=0

while status == "Running" or status == "Waiting":
    (msg,offset) =
        neos.getIntermediateResults(jobNumber,password,offset)
sys.stdout.write(msg.data)
    status = neos.getJobStatus(jobNumber, password)
time.sleep(2)

msg = neos.getFinalResults(jobNumber, password).data
sys.stdout.write(msg)
```

XML
Short-Job Queue

5-minute limit

- A few machines dedicated to this purpose
- Jobs exceeding limit are terminated

... prevents blocking of short jobs by long ones
New Solver-Submission Procedures

**Downloading the NEOS software**

**Installing the NEOS software**
- Client tools for problem submission
- **Solver tools for hooking up new solvers**
- Installing a new NEOS Server
Solver Submission

Hooking Up a New Solver

Register with NEOS

- Create an XML file to . . .
  - Describe your solver
  - Describe your solver’s input
  - Designate your workstation(s)
- Send the file to NEOS

Write a “driver” for your solver

Start a “server” for your solver on your workstation

Example: “HelloNEOS” solver . . .
Solver Submission

Describing Your Solver

<neos:SolverDescription xmlns:neos="http://www.mcs.anl.gov/neos">
  <neos:category>test</neos:category>
  <neos:solver>HelloNEOS</neos:solver>
  <neos:inputMethod>basic</neos:inputMethod>
  <neos:password>hello</neos:password>
  <neos:contact>fakeperson@mcs.anl.gov</neos:contact>
</neos:SolverDescription>
Describing the Solver’s Input

*Input types available*

- Text field
  - one line of text
- Text area
  - multiple lines of text
- File
  - name of a local file
- Check box
- Radio button

*Example continued . . .*
**Solver Submission**

**Describing the Solver’s Input**

```xml
<neos:input TYPE="textfield">
    <neos:token>num1</neos:token>
    <neos:filename>num1</neos:filename>
    <neos:prompt>First Number</neos:prompt>
</neos:input>

<neos:input TYPE="textfield">
    <neos:token>num2</neos:token>
    <neos:filename>num2</neos:filename>
    <neos:prompt>Second Number</neos:prompt>
</neos:input>

<neos:input TYPE="radio">
    <neos:token>operation</neos:token>
    <neos:filename>operation</neos:filename>
    <neos:prompt>Which Operation</neos:prompt>
    <neos:option value="Multiplication" default="true">Multiplication</neos:option>
    <neos:option value="Addition">Addition</neos:option>
</neos:input>
```
Solver Submission

Designating Workstations

```xml
<neos:machine>
  <neos:hostname>lully.mcs.anl.gov</neos:hostname>
  <neos:user>neos</neos:user>
</neos:machine>
</neos:SolverDescription>
```

Registering with NEOS

```
register.py HelloNEOS.txt
```
Solver Submission

Writing a “Driver” for the Solver

```python
#!/usr/bin/env python
import os
print ("Hello NEOS!");

f = open('num1','r')
num1 = float(f.read())
f.close()

f = open('num2','r')
num2 = float(f.read())
f.close()

f = open('operation','r')
operation=f.read()
f.close()

if operation=='Multiplication':
    print "%.5f * %.5f = %.5f" % (num1, num2, num1*num2)
else:
    print "%.5f + %.5f = %.5f" % (num1, num2, num1+num2)
```
Starting a “Server” on the Workstation

**List solvers in** /home/neos/driverlist.txt
- test:HelloNEOS:basic /path/to/hello.py

**Edit SolverTools/config.py**
- class Variables:
  - NEOS_HOST="neos.mcs.anl.gov"
  - NEOS_PORT=3332
  - JOBSDIR="/home/neos/HelloNEOS/jobs"
  - LOGDIR="/home/neos/HelloNEOS/logs"
  - TESTDIR="/home/neos/HelloNEOS/test"
  - DRIVER_FILE="/home/neos/driverlist.txt"

**Start up**
- SolverTools/SolverDaemon.py

**Open up a port (if behind a firewall)**
- SolverDaemon.py 4000
Optimization Services

Decentralized framework
- Centralized repository, but decentralized . . .
- Sources for solver information and problem analysis
- Services for optimization

Optimization cyberinfrastructure
- XML-based standards for representation
- Benchmarking and verification services
- High-performance computing on demand

What’s special about optimization?
- Independence of modeling, data, and solver software
- Choice of solver based on mathematics
- Huge variation in solver performance
**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.optimizationservices.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
<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 link=#0000cc 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></p></center></body></html>
```

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

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
Standards

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

```xml
<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>
```
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

```xml
<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>
```
### Example in OSiL (continued)

```xml
<linearConstraintCoefficients numberOfValues="3">
  <start>
    <el>0</el>
    <el>2</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)

```
<nl idx="1">
  <ln>
    <times>
      <variable idx="0"/>
      <variable idx="1"/>
    </times>
  </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
- 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
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
Translating from a Modeling Language

Sample model in AMPL

```AMPL
set ORIG;  # origins
set DEST;  # destinations

param supply {ORIG} >= 0;  # amounts available at origins
param demand {DEST} >= 0;  # amounts required at destinations

param vcost {ORIG,DEST} >= 0;  # variable shipment costs per unit
param limit {ORIG,DEST} > 0;  # limit on units shipped
var Trans {ORIG,DEST} >= 0;  # units to ship

param fcost {ORIG} >= 0;       # fixed costs for use of origins
var Use {ORIG} binary;         # = 1 iff origin is used

minimize Total_Cost:
  sum {i in ORIG, j in DEST}
    vcost[i,j] * Trans[i,j] / (1 - Trans[i,j]/limit[i,j]) +
  sum {i in ORIG} fcost[i] * Use[i];

subject to Supply {i in ORIG}:
  sum {j in DEST} Trans[i,j] <= supply[i] * Use[i];

subject to Demand {j in DEST}:
  sum {i in ORIG} Trans[i,j] = demand[j];
```
Translating from AMPL (cont’d)

AMPL data

```
param: ORIG:  supply fcost:=
    GARY 2800 50000  CLEV 5200 3940000  PITT 5800 370000 ;
param: DEST:  demand :=
    FRA 900  DET 1200  LAN 600
    WIN 400  STL 1700  FRE 1100
    LAF 1000 ;
param vcost :  FRA  DET  LAN  WIN  STL  FRE  LAF :=
    GARY  39  14  11  14  16  82   8
    CLEV  27  9   12  9   26  95   17
    PITT  24  14  17  13  28  99   20 ;
param limit :  FRA  DET  LAN  WIN  STL  FRE  LAF :=
    GARY 1300 2500 1500 1500 1900 1700 1700
    CLEV 800  900  800  800  500  700  1700
    PITT 2600 1700 2600 2700 1700 1900 2700 ;
```
Data Structures

Translating from AMPL (cont’d)

AMPL session

```
ampl: model nltrans.mod;
ampl: data nltrans.dat;

ampl: option solver amplclient;
ampl: option amplclient_options "solver lindo";
ampl: option lindo_options "...";

ampl: solve;

LINDO 12.1
LOCALLY OPTIMAL SOLUTION FOUND ...

ampl: display Trans;

...```
Translating from AMPL (cont’d)

AMPL’s “nl” output format (beginning)

```
g3 2 1 0   # problem nltrans
24 10 1 0 7 # vars, constraints, objectives, ranges, eqns
0 1        # nonlinear constraints, objectives
0 0        # network constraints: nonlinear, linear
0 21 0     # nonlinear vars in constraints, objectives, both
0 0 0 1    # linear network variables; functions; arith, flags
3 0 0 0 0   # discrete vars: binary, integer, nonlinear (b,c,o)
45 24       # nonzeros in Jacobian, gradients
0 0         # max name lengths: constraints, variables
0 0 0 0 0   # common exprs: b,c,o,c1,o1

C0         #Supply['GARY']
n0
C1         #Supply['CLEV']
n0
C2         #Supply['PITT']
n0
C3         #Demand['FRA']
n0
C4         #Demand['DET'] ...
```

Data Structures
Data Structures

Translating from AMPL (cont’d)

AMPL’s “nl” output format (nonlinear objective)

```
O0  0  #Total_Cost
o54  #sumlist
21
o3  #/
o2  #*
39
v0  #Trans['GARY','FRA']
o1  # -
1
o3  #/
v0  #Trans['GARY','FRA']
n1300
o3  #/
o2  #*
4
v1  #Trans['GARY','DET']
o1  # -
1
o3  #/
v1  #Trans['GARY','DET']
2500 ...
```
Translating from AMPL (cont’d)

OSiL derived from AMPL’s output format

```
<osil xmlns="os.optimizationservices.org"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="os.optimizationservices.org ../schemas/OSiL.xsd">
   <instanceHeader>
      <description>Generated from AMPL nl file</description>
   </instanceHeader>
   <instanceData>
      <variables numberOfVariables="24">
         <var name="_svar[1]"/>
         <var name="_svar[2]"/>
         ...
         <var name="_svar[22]" type="B" ub="1"/>
         <var name="_svar[23]" type="B" ub="1"/>
         <var name="_svar[24]" type="B" ub="1"/>
      </variables>
      ...
   </instanceData>
</osil>
```
Data Structures

Translating from AMPL (cont’d)

OSiL derived from AMPL’s output format

```xml
<objectives numberOfObjectives="1">
    <obj maxOrMin="min" numberOfObjCoef="24">
        <coef idx="21">50000</coef>
        <coef idx="22">3.94e+06</coef>
        <coef idx="23">370000</coef>
    </obj>
</objectives>

<constraints numberOfConstraints="10">
    <con name="_scon[1]" ub="-0"/>
    <con name="_scon[2]" ub="-0"/>
    <con name="_scon[3]" ub="-0"/>
    <con name="_scon[4]" lb="900" ub="900"/>
    <con name="_scon[5]" lb="1200" ub="1200"/>
    <con name="_scon[6]" lb="600" ub="600"/>
    <con name="_scon[7]" lb="400" ub="400"/>
    <con name="_scon[8]" lb="1700" ub="1700"/>
    <con name="_scon[9]" lb="1100" ub="1100"/>
    <con name="_scon[10]" lb="1000" ub="1000"/>
</constraints>
```
Data Structures

Translating from AMPL (cont’d)

OSiL derived from AMPL’s output format

```
<linearConstraintCoefficients numberOfValues="45">
  <start>
    <el>0</el>
    <el>2</el>
    <el>4</el>
    ...
  </start>
  <rowIdx>
    <el>0</el>
    <el>3</el>
    <el>0</el>
    <el>4</el>
    <el>0</el>
    <el>5</el>
    ...
  </rowIdx>
  <value>
    <el>1</el>
    ...
  </value>
</linearConstraintCoefficients>
```
Translating from AMPL (cont’d)

OSiL derived from AMPL’s output format

```
<linearConstraintCoefficients numberOfValues="45">
  <start>
    ...
  </start>
  <rowIdx>
    ...
  </rowIdx>
  <value>
    <el>1</el>
    <el>1</el>
    <el>1</el>
    <el>1</el>
    <el>1</el>
    ...
    <el>-2800</el>
    <el>-5200</el>
    <el>-5800</el>
  </value>
</linearConstraintCoefficients>
```
Data Structures

Translating from AMPL (cont’d)

OSiL derived from AMPL’s output format

```xml
<nonlinearExpressions numberOfNonlinearExpressions="1">
  <nl idx="-1">
    <sum>
      <divide>
        <times>
          <number value="39" type="real"/>
          <variable idx="0" coef="1"/>
        </times>
        <minus>
          <number value="1" type="real"/>
          <divide>
            <variable idx="0" coef="1"/>
            <number value="1300" type="real"/>
          </divide>
        </minus>
      </divide>
    </sum>
    ...
  </nl>
</nonlinearExpressions>
```
Data Structures

Translating from AMPL (cont’d)

OSrL derived from solver’s results

```xml
<osrl xmlns:os="os.optimizationservices.org"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="os.optimizationservices.org
                        ../schemas/OSiL.xsd">

  <resultHeader>
    <generalStatus type="success"/>
    <serviceName>Solved using a LINDO service</serviceName>
  </resultHeader>

  <resultData>
    <optimization numberOfSolutions="1" numberOfVariables="24"
                   numberOfConstraints="10" numberOfObjectives="1">
      ...
    </optimization>
  </resultData>

</osrl>
```
Data Structures

Translating from AMPL (cont’d)

OSrL derived from solver’s results

```xml
<solution objectiveIdx="-1">
  <status type="optimal"/>
  <variables>
    <values>
      <var idx="0">36.8552</var>
      <var idx="1">563.142</var>
      <var idx="2">122.355</var>
      <var idx="3">0</var>
      <var idx="4">991.065</var>
      ...
    </values>
    <other name="reduced costs">
      <var idx="0">0</var>
      <var idx="1">0</var>
      <var idx="2">0</var>
      <var idx="3">8.5573</var>
      <var idx="4">-2.51902e-09</var>
      ...
    </other>
  </variables>
</solution>
```
Data Structures

Translating from AMPL (cont’d)

OSrL derived from solver’s results

```
<objectives>
    <values>
        <obj idx="-1">722383</obj>
    </values>
</objectives>
<constraints>
    <dualValues>
        <con idx="0">-12.4722</con>
        <con idx="1">-98.9784</con>
        <con idx="2">0</con>
        <con idx="3">35.7967</con>
        <con idx="4">35.7967</con>
        <con idx="5">25.5129</con>
        <con idx="6">17.9149</con>
        <con idx="7">82.3857</con>
        <con idx="8">193.978</con>
        <con idx="9">29.3393</con>
    </dualValues>
</constraints>
</solution>
```
For More Information


- NEOS Server: www-neos.mcs.anl.gov/neos
