Cloud Pioneers:
NEOS and Optimization Services

Robert Fourer

Industrial Engineering & Management Sciences
Northwestern University, Evanston, IL, USA

AMPL Optimization LLC

4er@northwestern.edu — 4er@ampl.com

INFORMS Roundtable: "OR in the Cloud"

Austin, November 6-7, 2010

Session 3

Abstract

Today's move to cloud computing was foreshadowed by research reported at a number of past INFORMS meetings. The NEOS Server, hosted at Argonne National Laboratory and representing a collaboration of many members of the optimization community, has offered "optimization as a service" for over a decade. Optimization Services is a more recent initiative that offers a unified framework for distributed optimization over the Internet, including a set of XML-based protocols implemented by open-source software libraries (available at COIN-OR). We review both of these efforts, and the experience gained solving tens of thousands of optimization problems each month with this pioneering "OR in the cloud."

Motivation: Optimization Challenges

No one way to optimize

- Numerous problem classes
- Alternative methods for each class
- Competing free and commercial *solvers*

Models built to order

- Competing *modeling systems*
- Each system supports multiple solvers
- Many solvers work with multiple systems

A tangle of software

- No comprehensive packages as in stats or simulation
- Performance varies greatly

. . . ***hence an opportunity***

Solution: Optimization as a Service

NEOS Server

- Central scheduler
- Distributed compute engines

Optimization services framework

- Central registry
- Distributed servers
- Interface standards

NEOS neos.mcs.anl.gov

Network Enabled Optimization System

- Guide
  - tutorials, case studies, test problems, FAQs
- Server
  - free Internet access to solvers

NEOS Server

Since 1995 . . .

- Hosted at Argonne National Laboratory (Illinois, USA)
- Developed through 5 major releases
  - many contributors @ Argonne, Northwestern & elsewhere
  - increasingly sophisticated as Web has matured
- 10-20,000 server submissions in a typical month

. . . ***has handled over 100,000***

A research project

- Currently free of charge
- Supported by grants & volunteer efforts
- ***Moving in December*** . . .

NEOS @ WID

Wisconsin Institutes for Discovery **(discovery.wisc.edu)**

- Wisconsin Institute for Discovery (public)
- Morgridge Institute for Research (private)

Key participants

- Michael Ferris
  - research theme leader, *optimization in biology & medicine*
  - coordinator of NEOS move
- Miron Livny
  - founder of the Condor distributed-computing project
  - coordinator of computing technology for WID

Design

Flexible architecture

- Central controller and scheduler machine
- Distributed solver sites

Standard 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 ***(Kestrel)***
  - 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.

. . . more at www-neos.mcs.anl.gov/neos/stories.html

Who Uses NEOS? *(academic)*

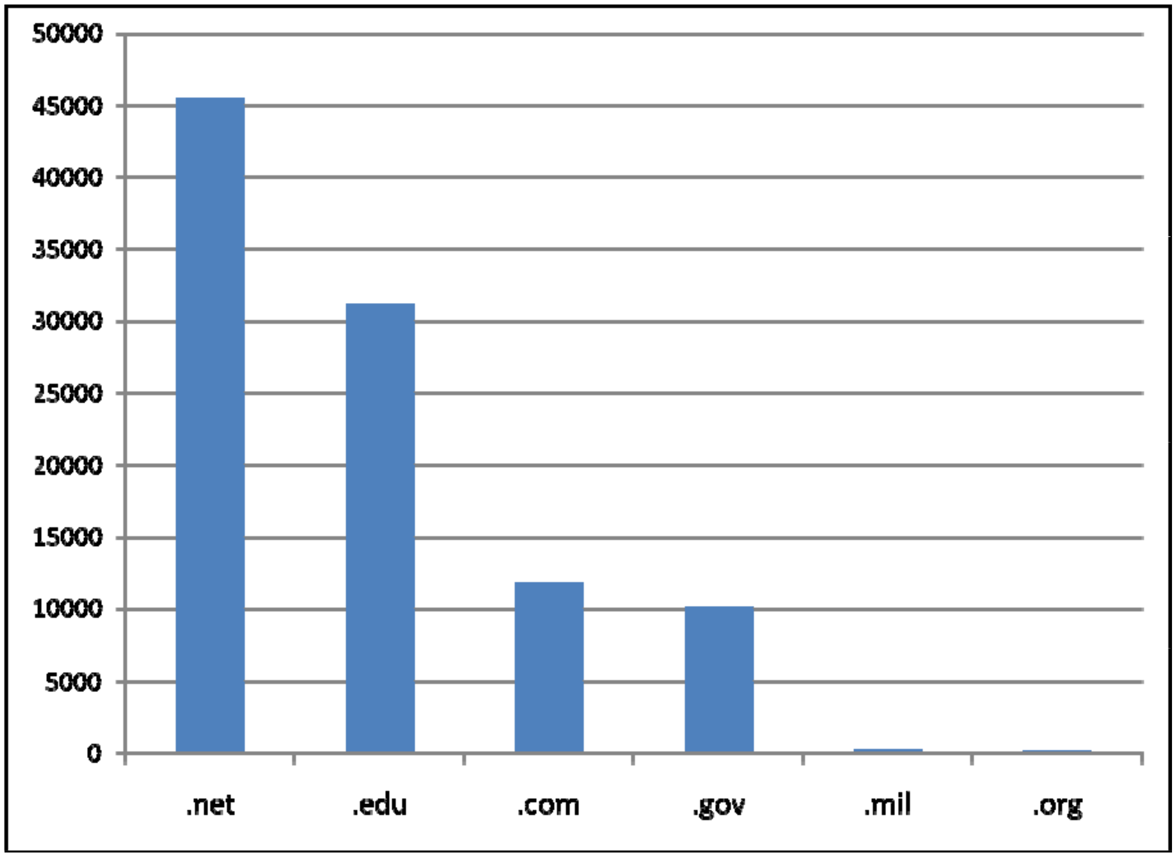
- I am regularly suggesting my students to use NEOS as soon as their projects in AMPL cannot be solved with the student edition. **So they debug their AMPL models locally . . . and then they run their real-life projects thanks to NEOS.**
- I didn't even know what nonlinear programming was and after I 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

NEOS Users

Where are They From?

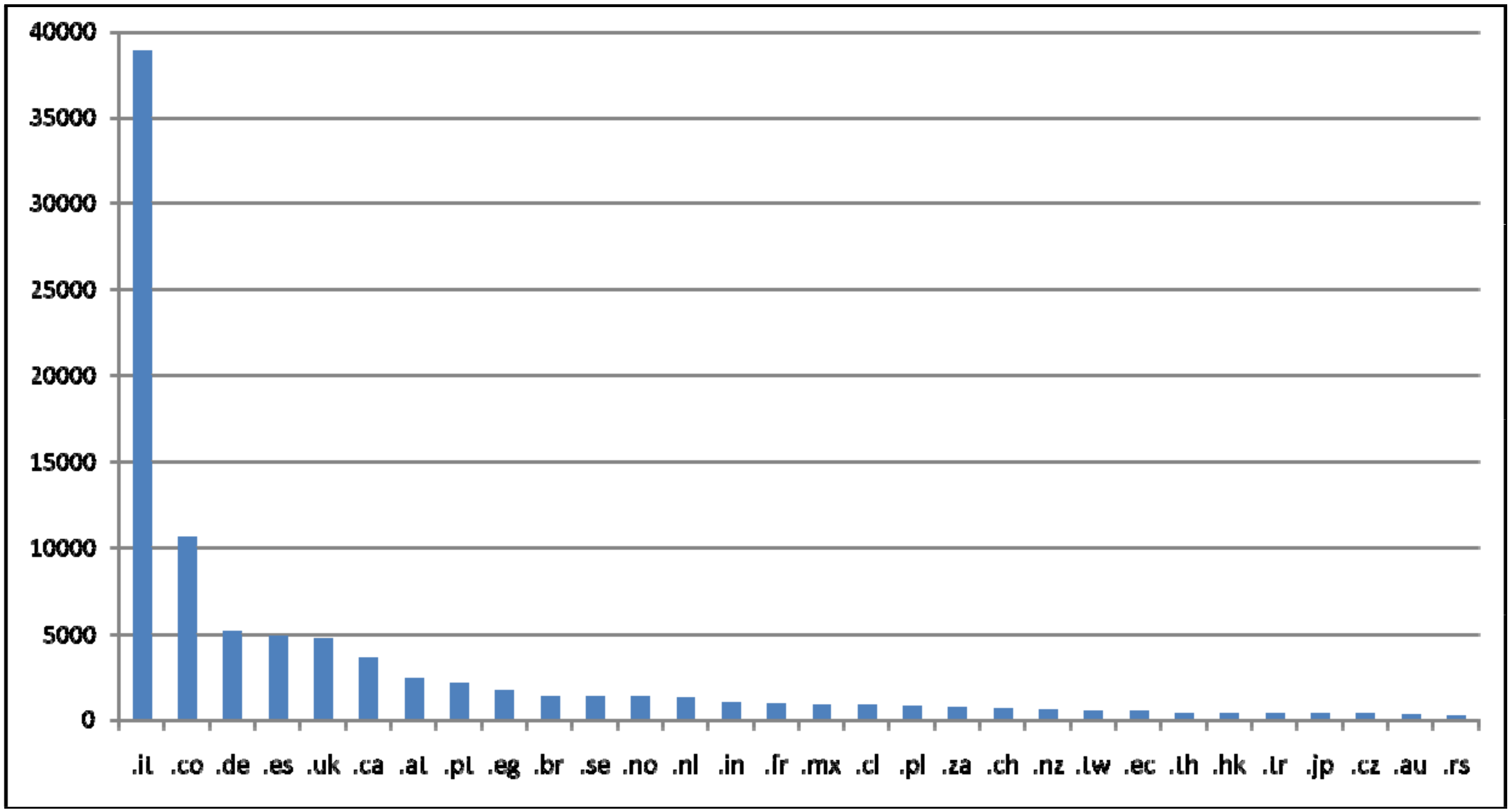
Standard domains



(2010 through October)

Where are They From?

Country domains (< 40000)

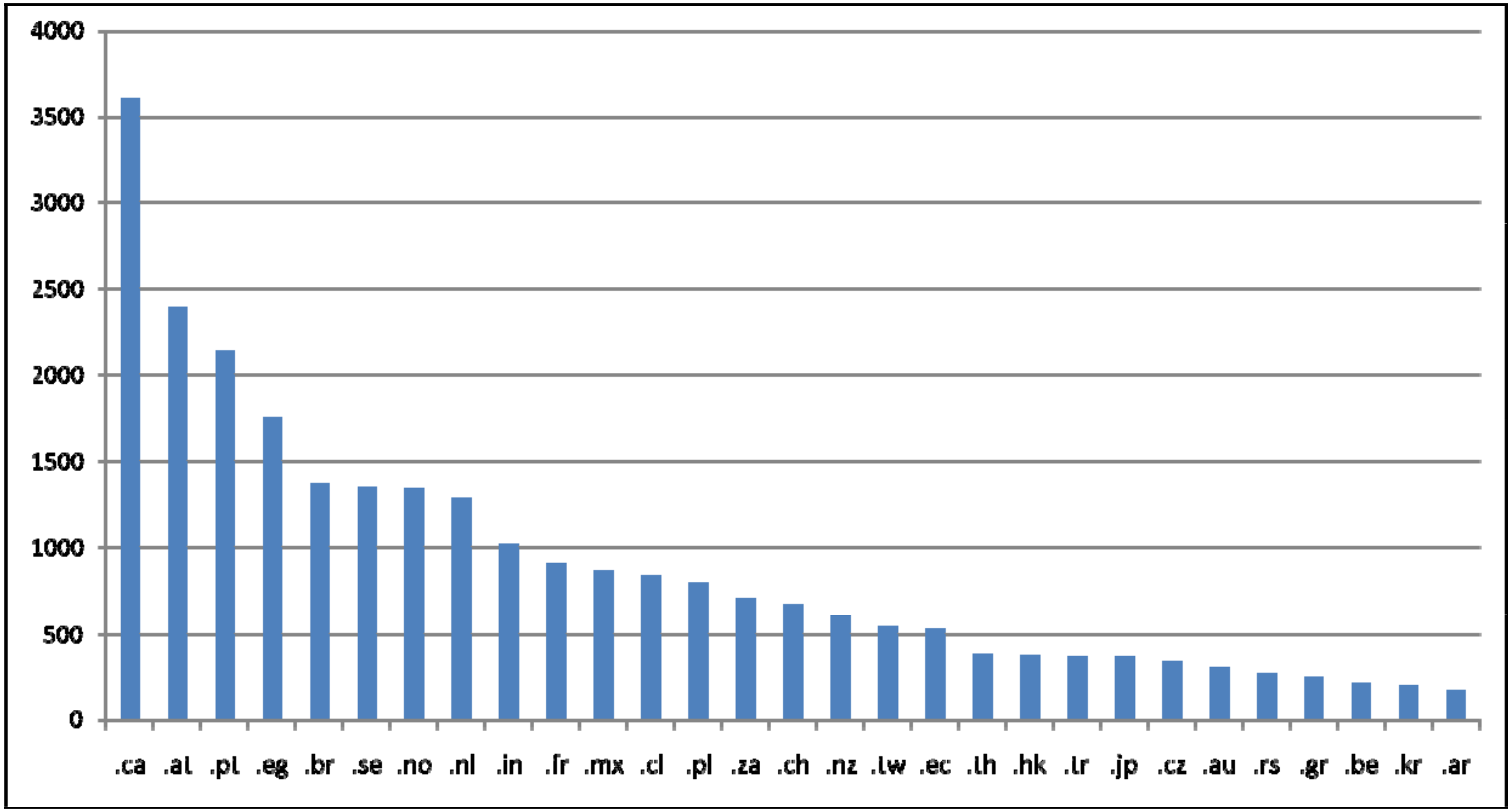


(2010 through October)

NEOS Users

Where are They From?

Country domains (< 4000)

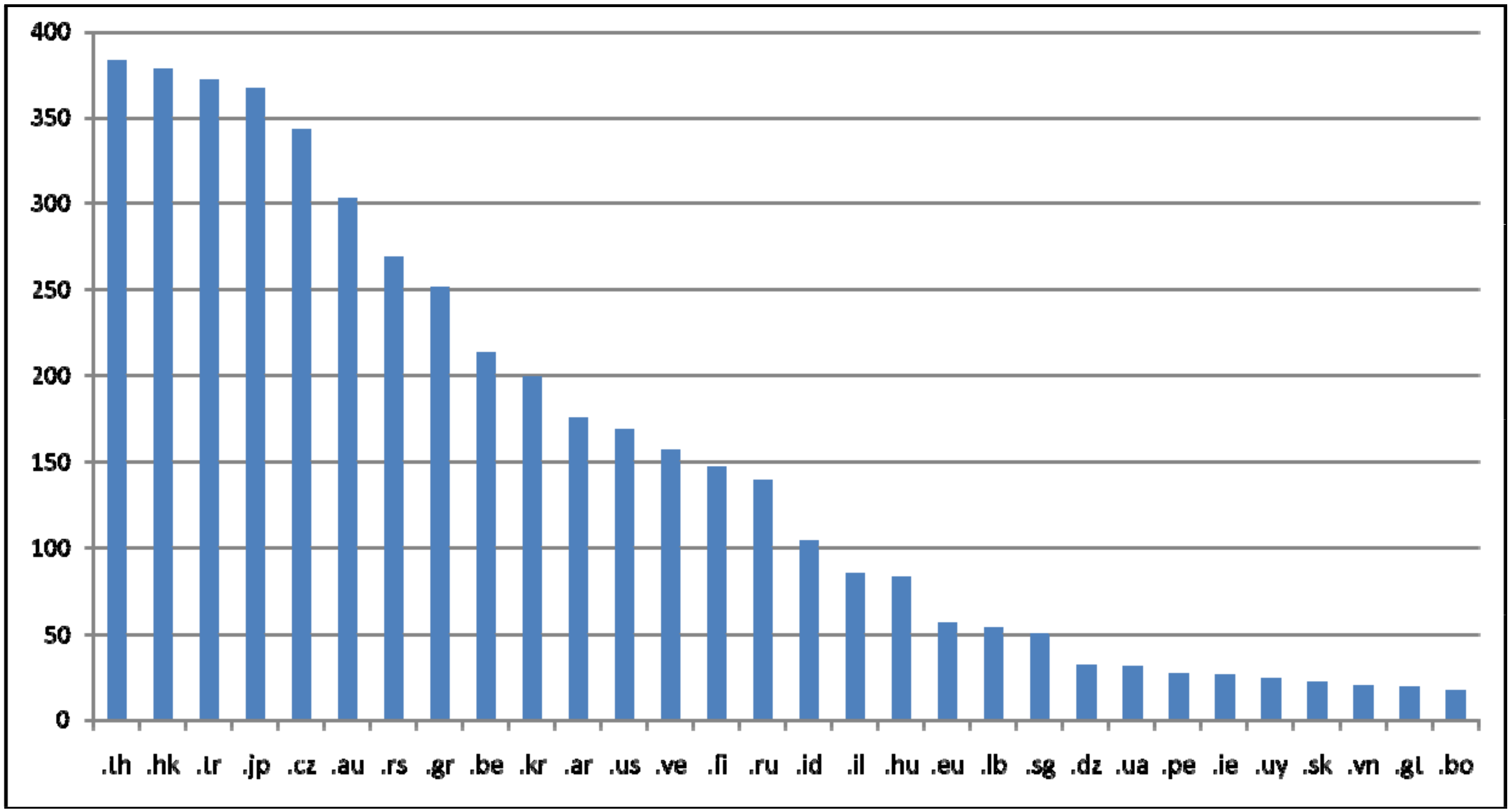


(2010 through October)

NEOS Users

Where are They From?

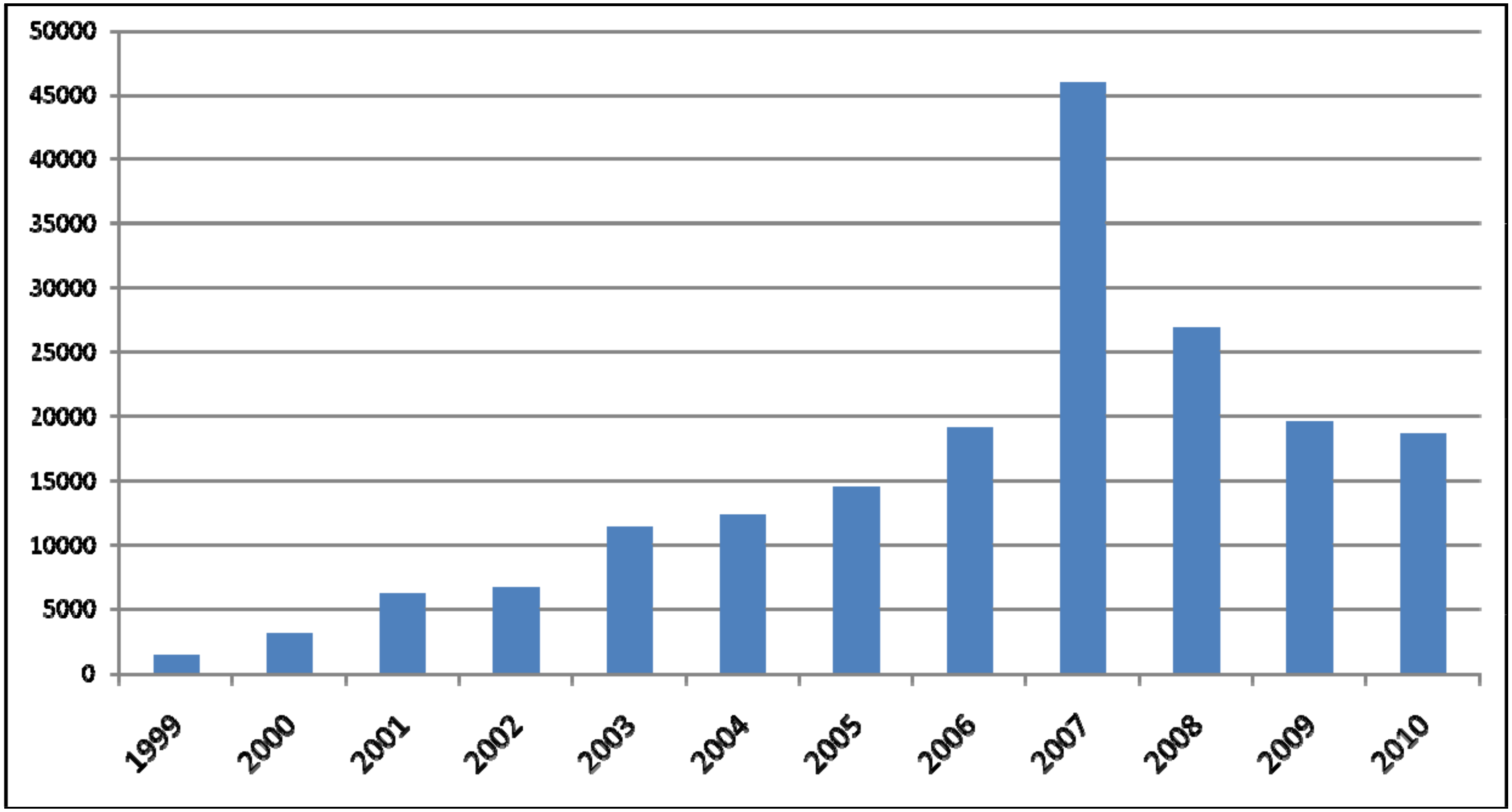
Country domains (< 400)



(2010 through October)

How Much Do They Use It?

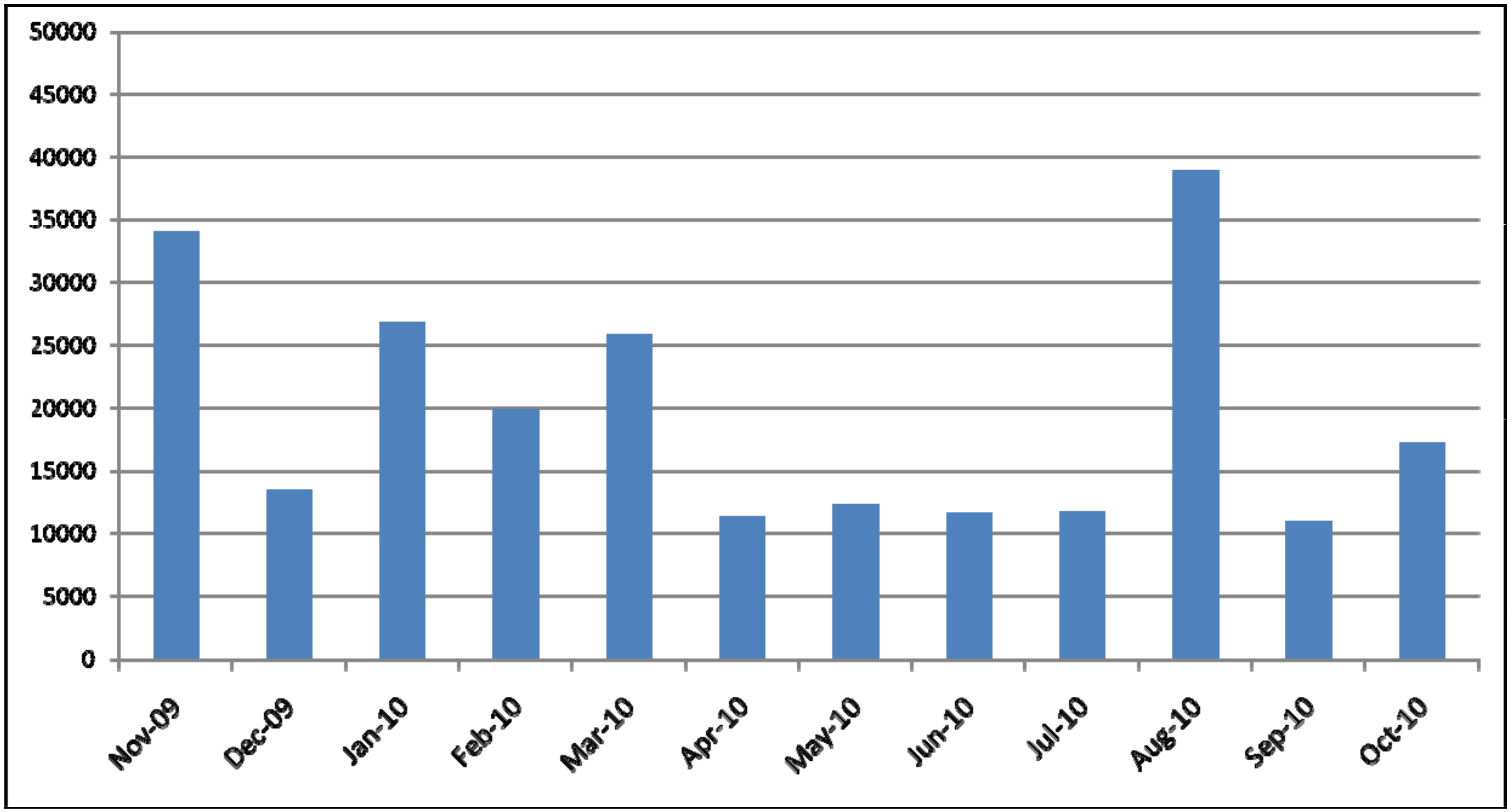
Monthly rates since 1999



20000/month ≈ 25/hour

How Much Do They Use It?

Monthly rates for past year



20000/month ≈ 25/hour

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

Who Supplies Them?

Some commercial solver vendors

- Xpress-MP, MOSEK, FortMP (mixed integer)
- CONOPT, KNITRO, MOSEK (nonlinear)

Universities and their researchers

- BonsaiG (mixed integer)
- DONLP2, LANCELOT, LOQO, MINOS, SNOPT (nonlinear)

Open-Source Enthusiasts

- GLPK, CBC, Bonmin (mixed integer)

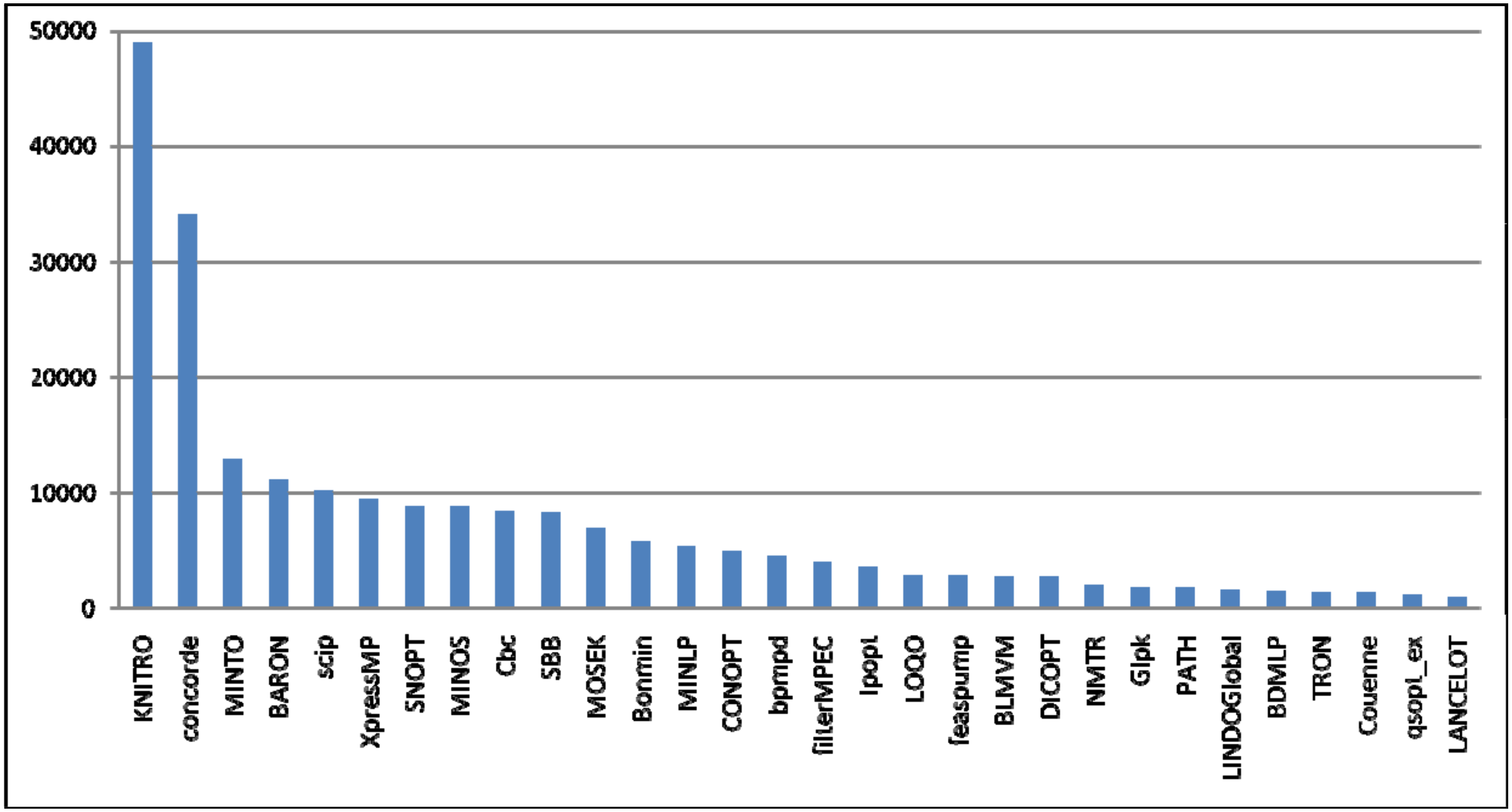
with thanks to . . .

- ***AMPL*** and ***GAMS*** developers
- ***Hans Mittelmann,*** Arizona State

NEOS Solvers

Which are Most Heavily Used?

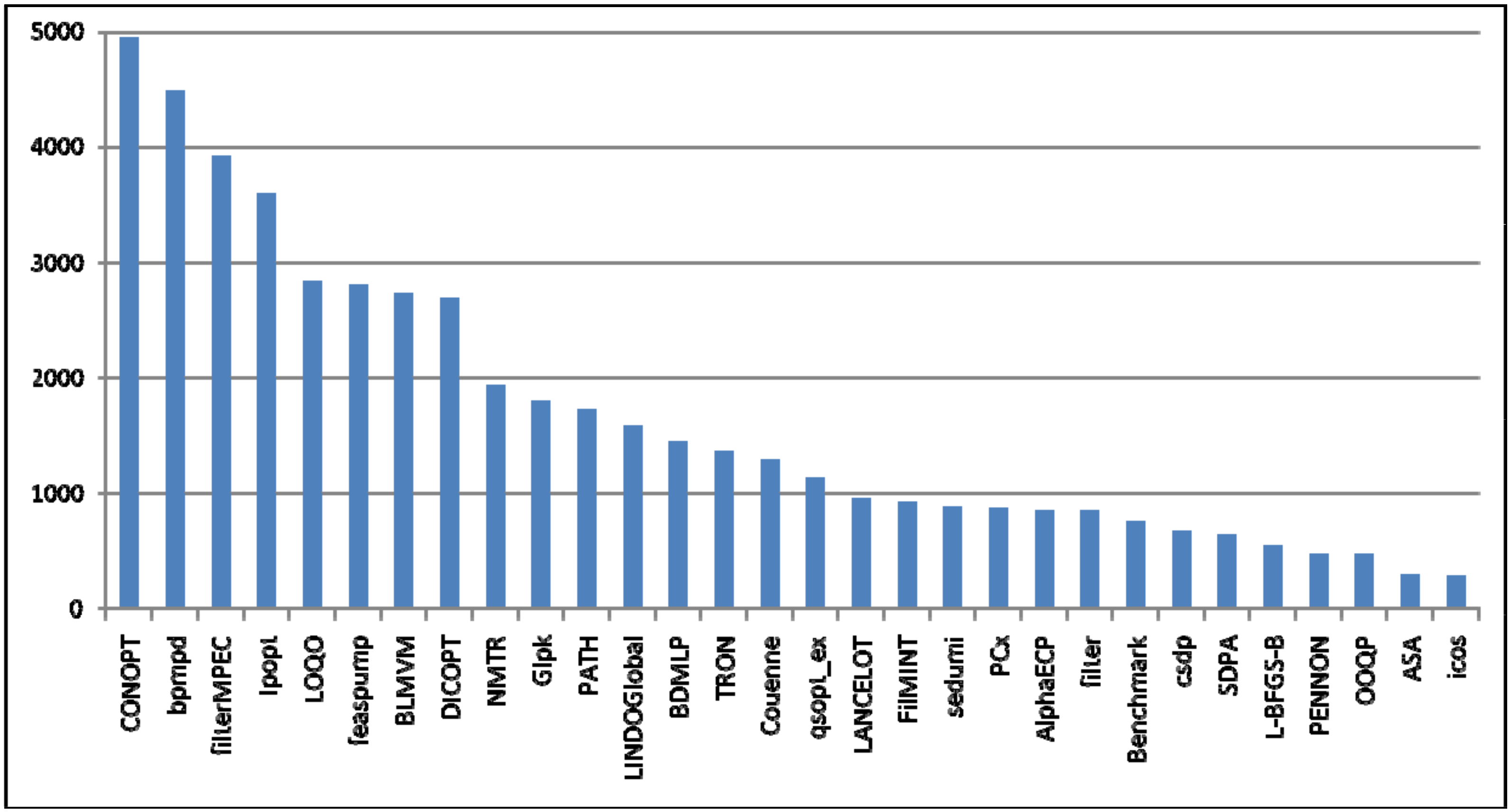
Solver submissions (< 50000)



(2010 through October)

Which are Most Heavily Used?

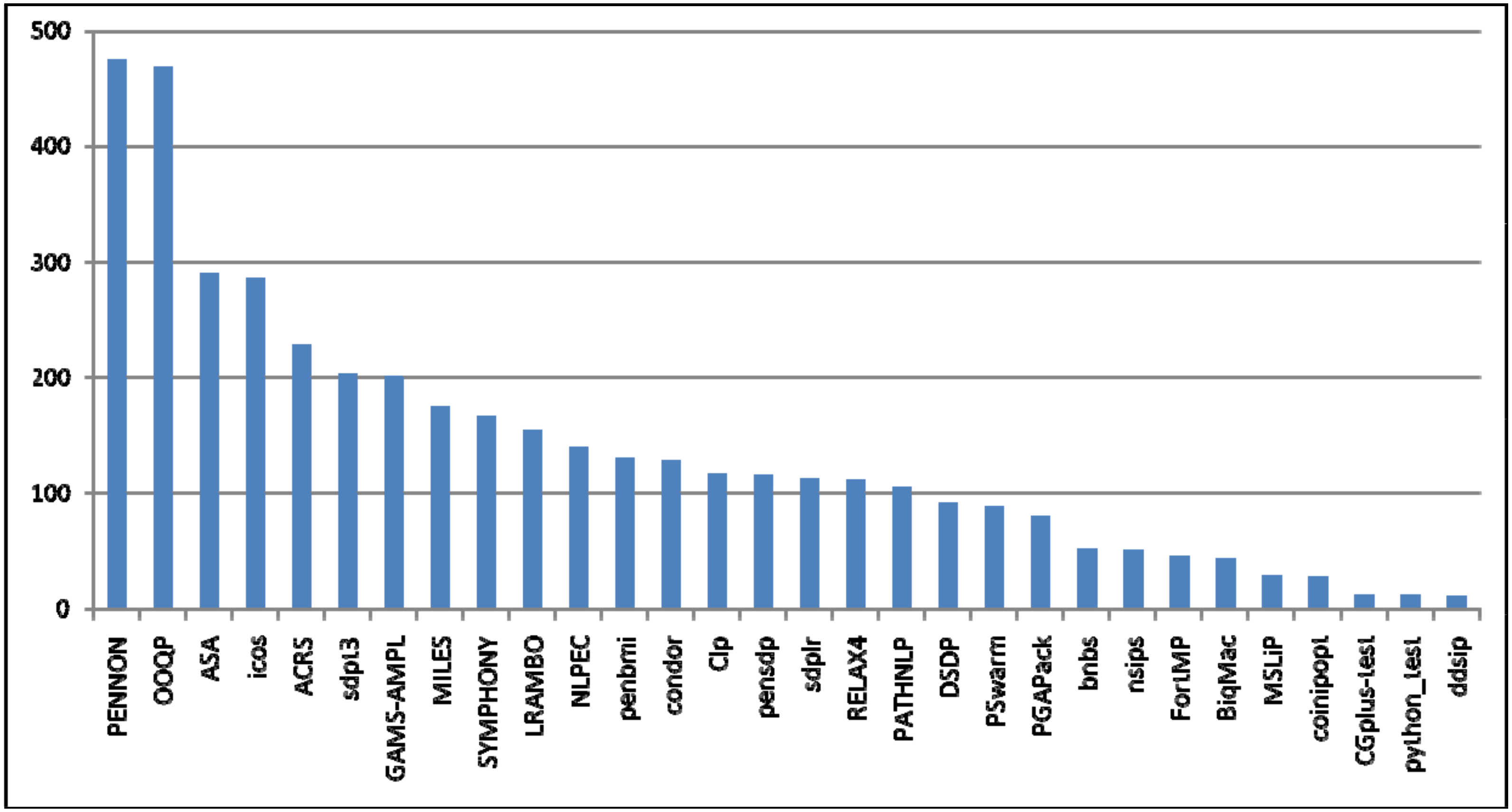
Solver submissions (< 5000)



(2010 through October)

Which are Most Heavily Used?

Solver submissions (< 500)



(2010 through October)

Where are They Hosted?

Varied workstations at

- Aachen University of Technology, Germany
- Argonne National Laboratory
- Arizona State University
- Lehigh University
- National Taiwan University
- Universidade do Minho, Portugal
- University of Wisconsin at Madison

. . . new hosts 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
- U.S. Department of Energy, Office of Advanced Scientific Computing, Mathematical, Information, and Computational Sciences Division subprogram, Contract W-31-109-Eng-38
- National Science Foundation, Challenges in Computational Science Program, grant CDA-9726385

Donations

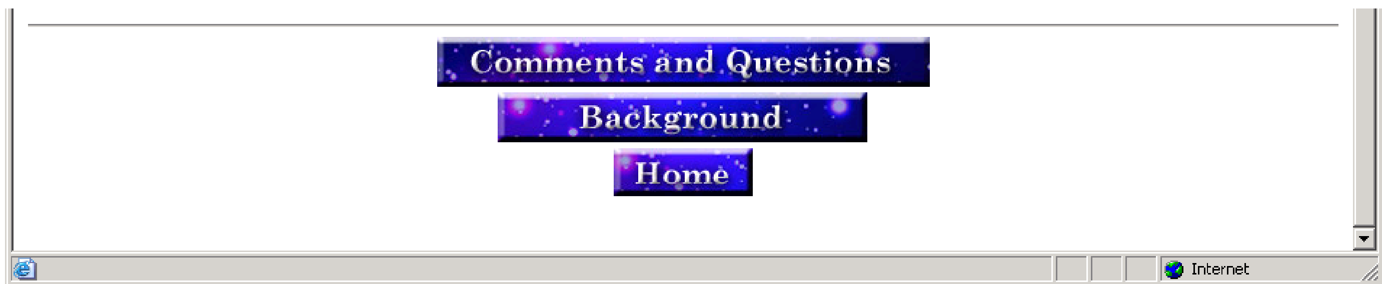
- Processor cycles
- Many people's time

Who Answers Users' Questions?

Large mailing list for support questions

- NEOS developers
- Solver developers

Support request buttons on every page



Notable Features

Callable interface

XML-based communications

Short-job queue

Solver submission procedures

Callable Interface

Server

- `http://neos.mcs.anl.gov:3332`
- at Argonne National Laboratory

Clients

- in Python, Perl, C, C++, Java, *and others*
- "Kestrel": clients for AMPL and GAMS

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)`

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>
```

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

```
#!/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: ...
```

XML

Example: Python Client *(cont'd)*

```
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)
```

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

Solver-Submission Procedures

Download

- **`www-neos.mcs.anl.gov/neos/Installation.html`**

Install

- Client tools for problem submission
- Solver tools for hooking up new solvers
- *Entire new Server installation*

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>
```

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 . . .

Describing the Solver's Input

```
<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>
```

Designating Workstations

```
<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
```

Writing a "Driver" for the Solver

```
#!/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`

NEOS Limitations

Limited choices for MIP

Limited input standardization

- Some AMPL, some GAMS
- Varied low-level formats

Limited support

- Maintenance
- Computing power

Limited funding model

- Grants?
- User fees?

. . . but forthcoming move may change things!

To Learn More . . .

Websites

- `neos.mcs.anl.gov`

Overview

- Elizabeth D. Dolan, Robert Fourer, Jorge J. Moré, and Todd S. Munson, "Optimization on the NEOS Server." SIAM *News* **35:**6 (July/August 2002) 4, 8–9. `www.siam.org/pdf/news/457.pdf`

AMPL/GAMS interface

- Elizabeth D. Dolan, Robert Fourer, Jean-Pierre Goux, Todd S. Munson and Jason Sarich, "Kestrel: An Interface from Optimization Modeling Systems to the NEOS Server." *INFORMS Journal on Computing* **20** (2008) 525–538. `dx.doi.org/10.1287/ijoc.1080.0264`

Optimization Services (OS)

A "next-generation NEOS"

- Decentralizes provider ***services***
- Adopts established web-service protocols
- Creates new ***standards*** for optimization

Origins

- Proposed XML standard for specifying LPs
  - ∗ Robert Fourer, Leo Lopes, Kipp Martin, "LPFML: A W3C XML Schema for Linear and Integer Programming." *INFORMS Journal on Computing* **17** (2005) 139–158.
- Jun Ma's thesis project
  - ∗ Jun Ma, "Optimization Services (OS)." Ph.D. dissertation, Northwestern University (2005).

OS

Development

- Open-source project hosted at COIN-OR
- Jun Ma & Kipp Martin, project directors
  - Gus Gassmann, Tim Middelkoop, Imre Pólik, Wayne Sheng

Distribution

- Source for Max OS X, Linux,
  numerous Windows configurations
- Binaries for most popular platforms

OS: Services

Registration

- OS project establishes centralized online registry
- Providers list their services with the registry

Discovery

- Prospective user queries the registry
- Registry reports appropriate solver services

Submission

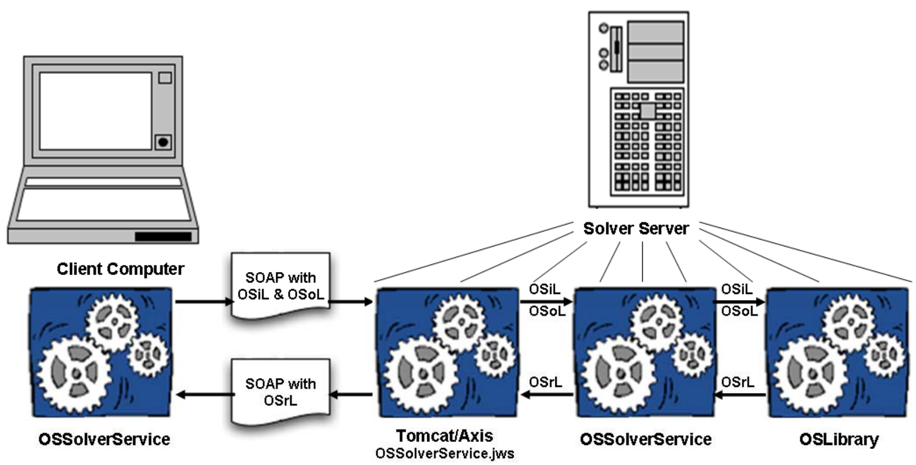
- User communicates directly with chosen services

. . . using new standards at every step

OS Service Software

OSSolverService

- Remote solver execution using web services
- Command-line or interactive guide
- Range of service methods
  - `solve`, `send`, `retrieve`, `getJobID`, `knock`, `kill`



OS Modeling Tools

OSAmplClient

GAMS CoinOS solver

- Call OSSolverService directly from AMPL or GAMS
  - ∗ Like NEOS's Kestrel,
    but remote location explicitly specified

OSmatlabSolver

- Convert MATLAB arrays to an OS problem instance
  - ∗ like converters for "nl" and "mps" forms
- Send to OSSolverService

OS Services Status

Current release 2.1

- Optimization Services 2.1 User's Manual
  - by Horand Gassmann, Jun Ma, Kipp Martin, Wayne Sheng

Still to come

- Registry
- Further integration with COIN-OR
  - Python modeling tools: PuLP, Pyomo
  - DIP decomposition framework
- Broader adoption outside of COIN-OR
  - More formal standards process

OS: Standards

Optimization instance representation *(XML)*

- problems (OSiL)
- solver directives (OSoL)
- solutions (OSrL)

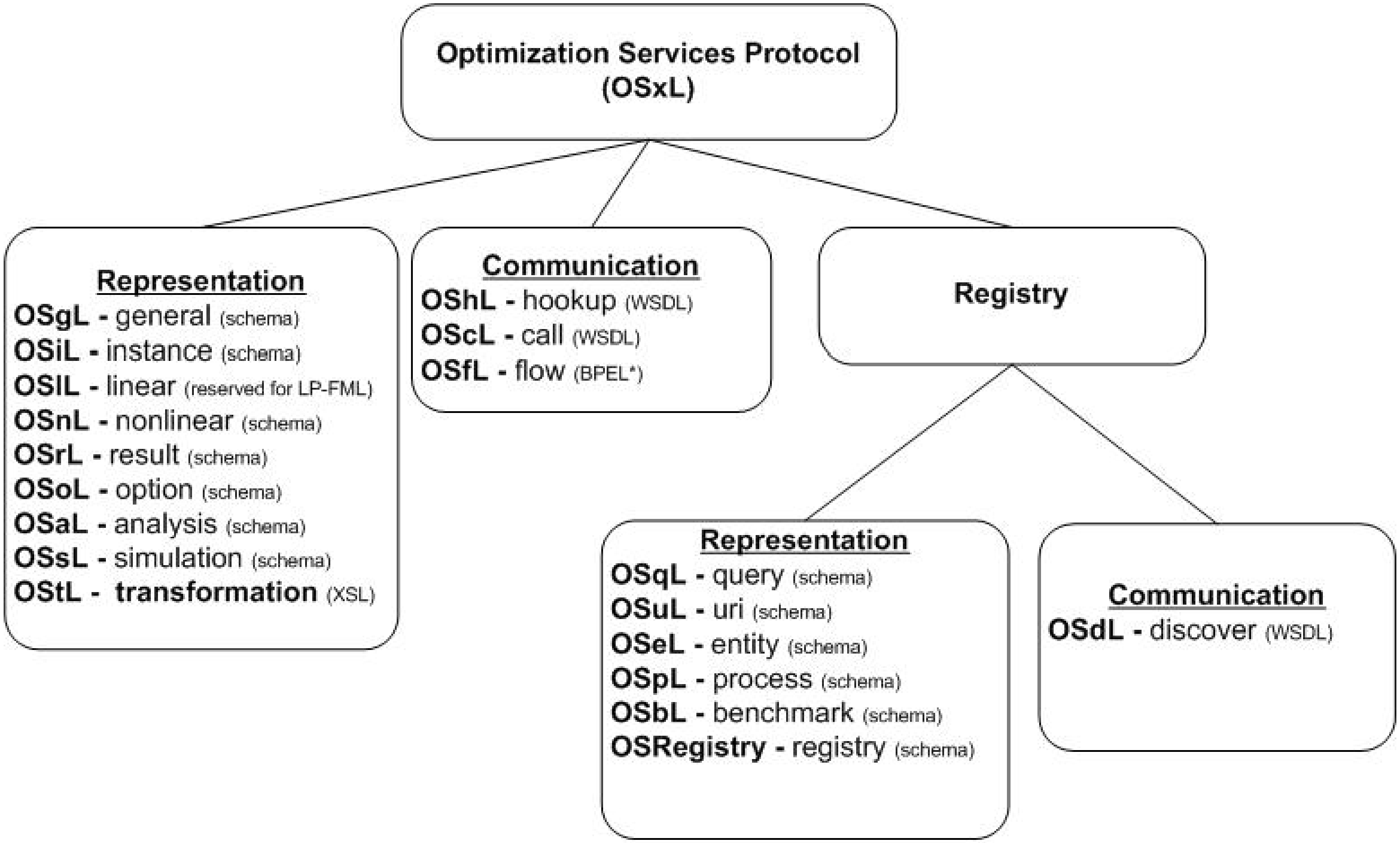
Optimization service registration & discovery *(XML)*

- solver entries (OSeL)
- registry queries (OSqL)
- problem analyses (OSaL)

Optimization communication *(WSDL)*

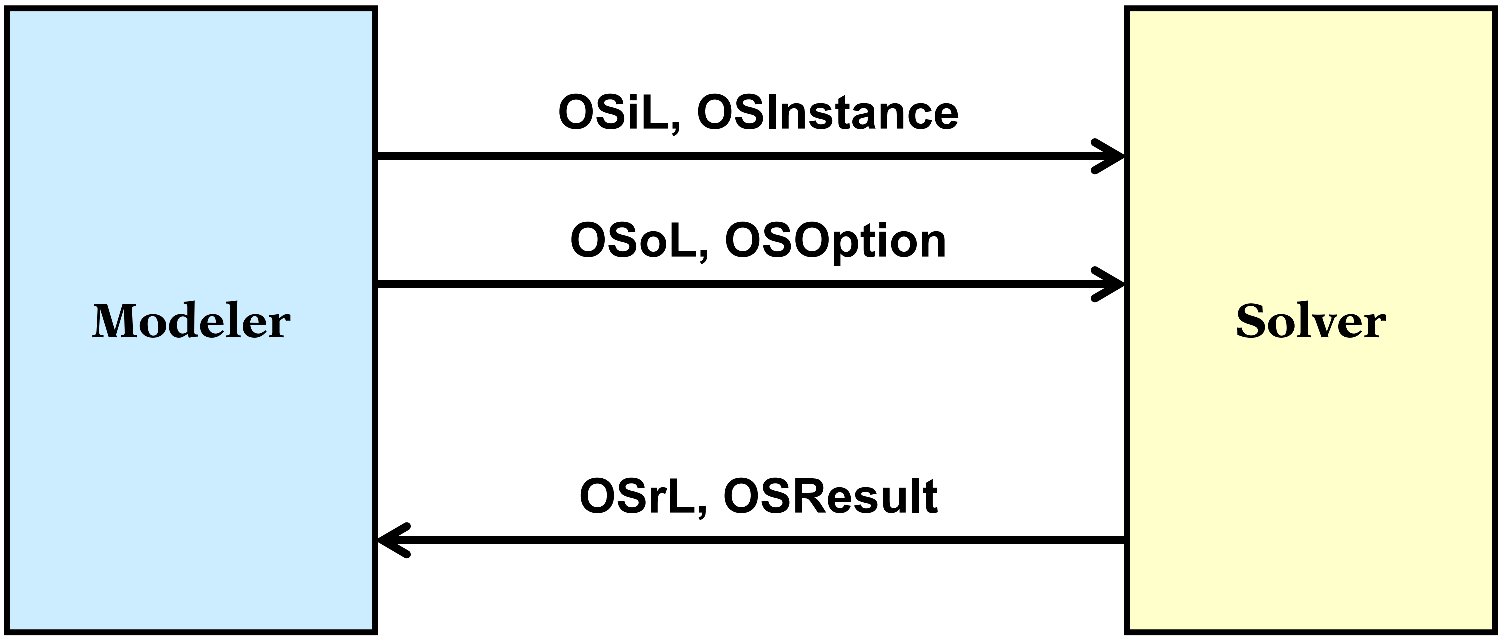
- accessing, interfacing, orchestration

Standards



***OSmL:** a modeling language and NOT an Optimization Services Protocol
***Letters not currently used:** w, z
***BPEL:** Business Process Execution Language for flow orchestration.

Problem Instance Standards



XML text files

- OSiL, OSoL, OSrL

In-memory data structures

- OSInstance, OSOption, OSResult

XML Means "Tagged" Text Files . . .

Example: html for a popular home page

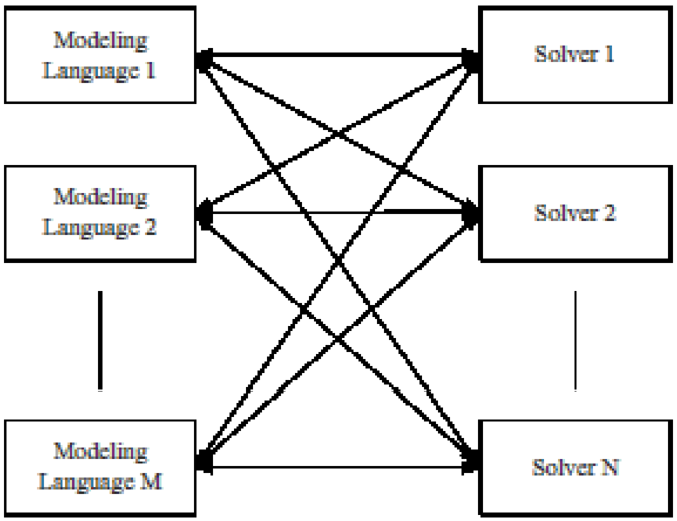
```
<html><head><meta http-equiv="content-type" content="text/html;
charset=UTF-8"><title>Google</title><style><!--
body,td,a,p,.h{font-family:arial,sans-serif;}
.h{font-size: 20px;}
.q{text-decoration:none; color:#0000cc;}
//-->
</style>
</head><body bgcolor=#ffffff text=#000000 link=#0000cc
vlink=#551a8b 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

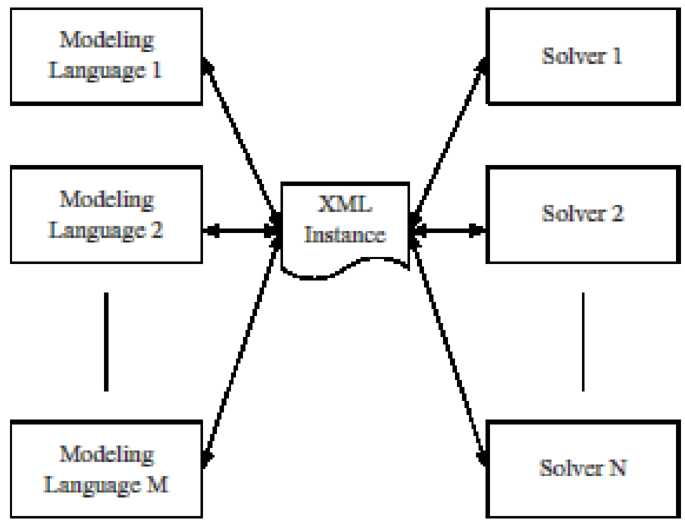
. . . by use of a schema written itself in XML

Advantage of any standard

MN drivers without a



M + N drivers with a standard



Motivation

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*

OSiL: Optimization Problem Instances

Design goals

- Simple, clean, extensible, object-oriented

Standard problem types supported

- Linear
- Quadratic
- General nonlinear
- Mixed integer
- Multiple objective
- Complementarity

OSiL *(cont'd)*

Extensions

- User-defined functions
- XML data (within the OSiL or remotely located)
- Data lookup (via XPath)
- Logical/combinatorial expressions and constraints
- Simulations (black-box functions)

OSiL *(cont'd)*

Prototypes

- Conic optimization
  - 2nd-order cone programming
  - semidefinite programming
- Stochastic programming
  - recourse, penalty-based, scenario (implicit or explicit)
  - risk measure/chance constrained
  - major univariate, multivariate, user-defined distributions
  - general linear transformation and ARMA processes

R. Fourer, H.I. Gassmann, J. Ma, and R.K. Martin,
"An XML-Based Schema for Stochastic Programs."
Annals of Operations Research **166** (2009) 313–337.

What about "MPS Form" / "LP 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**

Text files

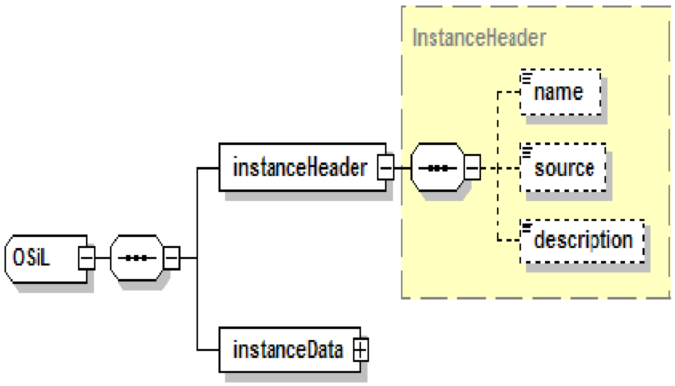
Text from the OSiL Schema

```
<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>
```

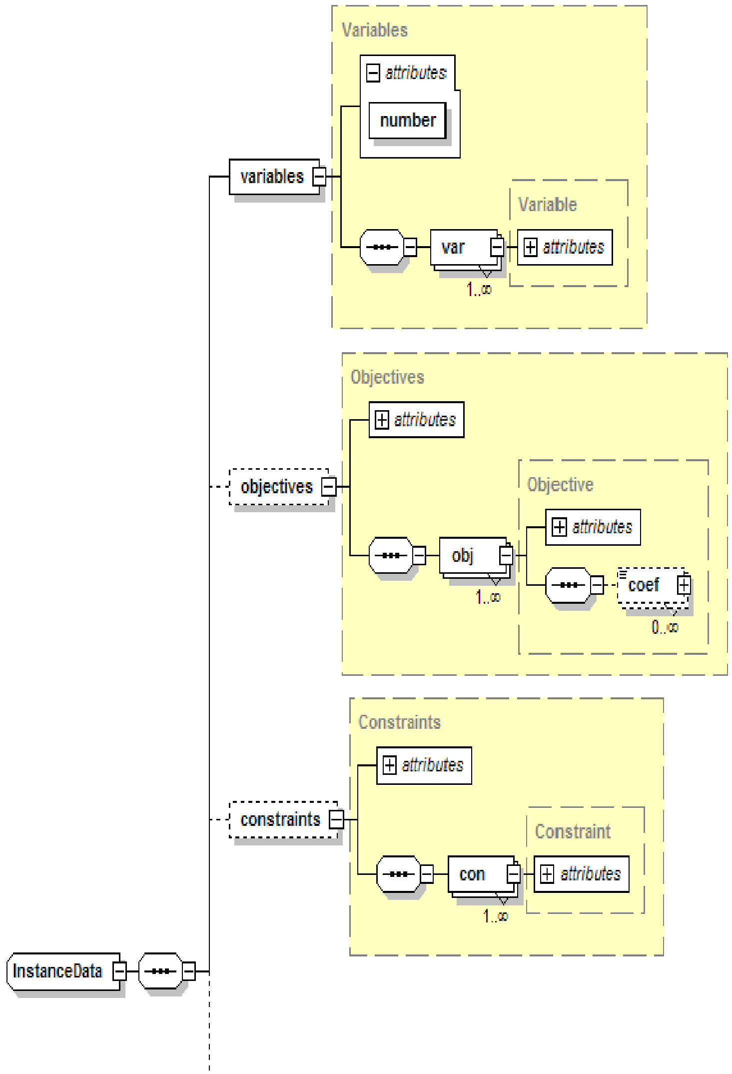
Text files

Diagram of the OSiL Schema



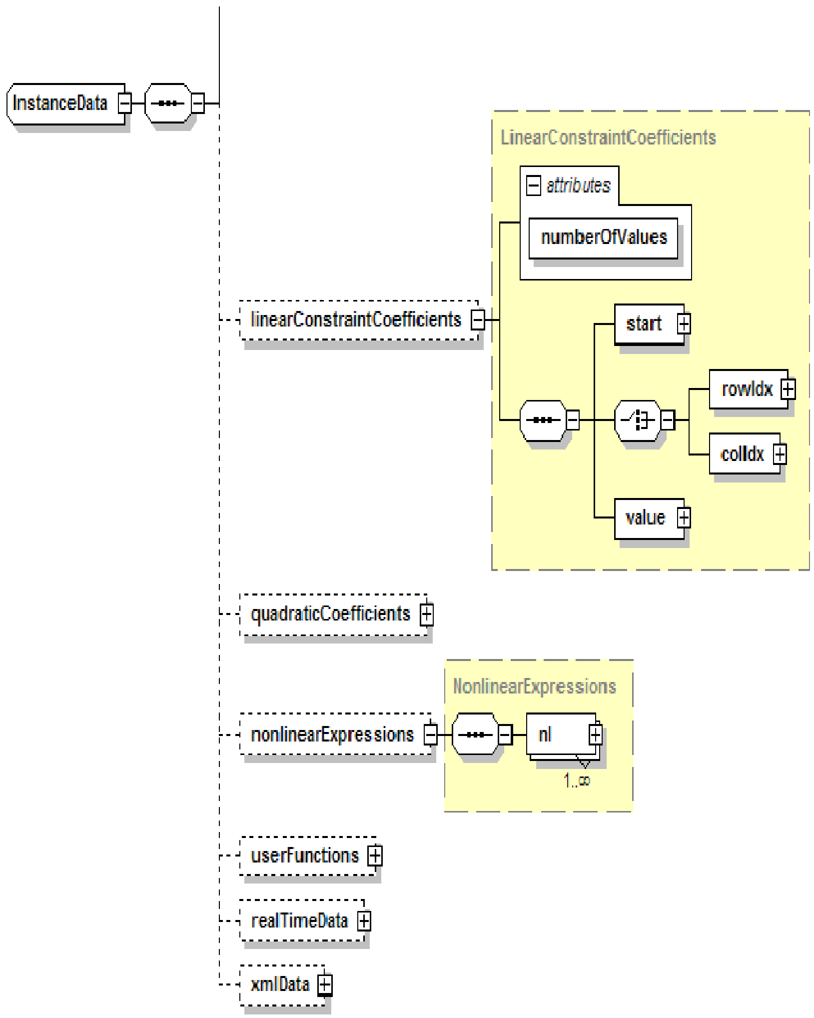
Text files

Details of OSiL's *instanceData* Element



Text files

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;
```

Text files

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>
```

Text files

Example in OSiL *(continued)*

```
<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>
```

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>
```

Example in OSiL *(continued)*

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

OSrL: Optimization Problem Results

Counterpart to OSiL for solver output

- General results such as serviceURI, serviceName, instanceName, jobID, time
- Results related to the solution such as status (unbounded, globallyOptimal, etc.), substatus, message
- Results related to variables (activities), objectives (optimal levels), constraints (dual values)
- Service statistics such as currentState, availableDiskspace, availableMemory, currentJobCount, totalJobsSoFar, timeLastJobEnded, etc.
- Results related to individual jobs including state (waiting, running, killed, finished), userName, submitTime, startTime, endTime, duration, dependencies, scheduledStartTime, requiredDirectoriesAndFiles.

OSrL *(cont'd)*

Additional solution support

- Support for non-numeric solutions
  such as those returned from
  combinatorial or constraint programming solvers
- Support for multiple objectives
- Support for multiple solutions
- Integration of analysis results
  collected by the solver

OSoL: Optimization Options

Counterpart to OSiL for solver instructions

- General options including serviceURI, serviceName, instanceName, instanceLocation, jobID, license, userName, password, contact
- System options including minDiskSpace, minMemorySize, minCPUSpeed
- Service options including service type
- Job options including scheduledStartTime, dependencies. requiredDirectoriesAndFiles, directoriesToMake, directoriesToDelete, filesToCreate, filesToDelete, processesToKill, inputFilesToCopyFrom, inputFilesToCopyTo, etc.

Limited standardization of algorithmic options

- Currently only initial values

OSoL *(cont'd)*

Including support for:

- Various networking communication mechanisms
- Asynchronous communication (such as specifying an email address for notification at completion)
- Stateful communication (achieved mainly through the built-in mechanism of associating a network request with a unique jobID)
- Security such as authentication and licensing
- Retrieving separately uploaded information (when passing a large file as a string argument is inefficient)
- Extended or customized solver-specific or algorithm-specific options

Other XML Schema-Based Standards

Kept by the OS registry

- OSeL (entity, experimental): static information on optimization services (such as type, developer)
- OSpL (process, near stable): dynamic information on optimization services (such as jobs being solved)
- OSbL (benchmark, experimental): benchmark information on optimization services

For use by the discovery process

- OSqL (query, experimental): specification of the query format used to discover the optimization services in the OS registry
- OSuL (uri/url, experimental): specification of the discovery result (in uri or url) sent back by the OS registry

Other Schema-Based Standards *(cont'd)*

Formats and definitions

- OSsL (simulation, stable): format for input and output used by simulation services invoked via the Optimization Services to obtain function values
- OSgL (general, near stable): definitions of general elements and data types used by other OSxL schemas. Usually included in the beginning of another OSxL schema through the statement:
  ```
  <xs:include schemaLocation="OSgL.xsd"/>
  ```
- OSnL (nonlinear, stable): definitions (operators, operands, etc.) of the nonlinear, combinatorial, and other nodes used in other OSxL's, mainly OSiL

Other WSDL-Based Standards

WSDL

- Web Service Definition Language

WSDLs for OS (stable)

- OShL (hook): for invoking solver/analyzer services
- OSdL (discover): for invoking optimization registry services to register and discover services
- OScL (call) for invoking simulation services, usually to obtain function values.

OS Standards Status

Instance standards

- OSiL well established
- OSrL completed
- OSoL near completion
- OSmL next
  - for specifying problem modifications

Registration & communications standards

- awaiting further development of registry

OS Limitations vs. NEOS

Limited choices for MIP

- Mostly the COIN-OR solvers

Full input standardization

Limited support

Limited funding model

To Learn More . . .

Websites

- **`www.optimizationservices.org`**
- **`projects.coin-or.org/OS`**

Overview

- Robert Fourer, Jun Ma, Kipp Martin, "Optimization Services: A Framework for Distributed Optimization." *Operations Research* ***58*** (2010) 1–13.
- Robert Fourer, Jun Ma and Kipp Martin, "OSiL: An Instance Language for Optimization." *Computational Optimization and Applications* **45** (2010) 181–203.

Guide

- Optimization Services 2.1 User's Manual: `www.coin-or.org/OS/doc/osUsersManual_2.1.pdf`
- Examples of use: `projects.coin-or.org/svn/CoinBazaar/projects/ApplicationTemplates`

To Learn More . . .

Talks at this conference

- **TD40: COIN-OR Under the Hood,**
  Kipp Martin, *COIN Easy*
- **WA40: Solver APIs II,**
  Kipp Martin, *The Optimization Services Solver Interface*

Implications for Cloud Computing

Need a complete solution

- Variety of modeling & solving products
- Assistance with selection

Optimization poses special challenges

- Highly uncertain run times
- Mixed software environments

Financial model is critical

- Charging for use
- Sharing revenues
- Supporting cooperative efforts