Hooking Optimization Services
to Modeling Languages and Solvers

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

INFORMS Annual Meeting
Seattle, Washington — 4-7 November 2007 — SD29.3

“Optimization Services” (OS)
A framework for providing optimization tools
- XML-based
- Service-oriented
- Distributed
- Decentralized

A project for implementing such a framework
- Straightforward and ubiquitous access
- Powerful solvers

Using a robust service-oriented architecture
- Linking modeling languages, solvers, schedulers, data repositories
- Residing on different machines, in different locations, using different operating systems.
Licensing

*Released as open source code*
- Under the Common Public License (“CPL”)

*Available as a COIN-OR project*
- *Version 1.0 released November 1, 2007!*

OS on the Internet

*Home site:* www.optimizationservices.org
- Overview, standards, publications, presentations, FAQs
- Contact information, downloads, licenses

*Developer site:* https://projects.coin-or.org/OS
- Wiki, source repository, timeline, register, login, search

*COIN-OR listing:* www.coin-or.org/projects/OS.xml
- License, language, development status
- Links to related COIN-OR and third-party software
OS on the Internet (cont’d)

Newsgroup:
groups.google.com/group/optimizationservices

COIN-OR mailing list:
list.coin-or.org/mailman/listinfo/os

... automatically cross-posted

OS Builds: Platforms

Unix
- Mac
- Linux

Windows
- Microsoft Windows (MS Visual Studio)
- Cygwin (gcc)
- MSYS (gcc, cl.exe)
OS Builds: Integration

Core (OSCommon library)

Modeler side
- AMPL / .nl
- LINGO, What's Best (planned)
- MATLAB

Solver side
- COIN OSI
- AMPL/ASL
- Linear: CLP, CBC, CPLEX, Impact
- Nonlinear: IPOPT, LINDO, KNITRO
- CppAD (automatic differentiation)

. . . looking for developers to provide others

The OSiL Standard

Purpose
- Represent optimization problem instances

Design goals
- Simple, clean, extensible, object-oriented

Scope
- All standard problem types supported
  - Linear
  - Quadratic
  - General nonlinear
  - Mixed integer for any of above
  - Multiple objective for any of above
  - Complementarity
  - Logic
  - . . .
**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)

**Prototypes**
- Cone & semidefinite programming
- Stochastic
  - recourse, penalty-based, scenario (implicit or explicit)
  - risk measure/chance constrained
  - major univariate, multivariate, user-defined distributions
  - general linear transformation and ARMA processes

**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:
  ```xml
  <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.
www.optimizationservices.org . . .

. . . Questions?