Recent Developments in Optimization Services

H.I. Gassmann, Dalhousie University
J. Ma, Northwestern University
R.K. Martin, The University of Chicago

Optimization Days, Montreal, May 2011
Outline

• Distributed computing and OR
• Optimization Services
• Solver options
• OSoL – OS option language
• Solver results
• OSrL – OS result language
• Availability
OR development cycle

• Model building
• Data collection
• Instance generation
• Problem solution
• Result analysis
• …potentially all on different computers
What Is Optimization Services (OS)?

• Web-aware framework that connects algebraic modelling languages and optimization solvers

• XML-based standards for representing optimization instances (OSiL), optimization results (OSrL), optimization solver options (OSoL), etc.

• Open source libraries that implement the standards (under COIN-OR)

• A robust API for both solver algorithms and modeling systems

• A command line executable OSSolverService

• OSAmplClient, an executable to work with the AMPL modeling language

• Utilities that convert MPS files and AMPL nl files into OSiL

• Server software that works with Apache Tomcat and Apache Axis
Why Optimization Services?

Optimization services is needed because there is/are:

• Numerous modeling languages each with their own format for storing the underlying model.
• Numerous solvers each with their own application program interface (API).
• Numerous operating system, hardware, and programming language combinations.
• No standard for representing problem instances, especially nonlinear optimization instances.
• No real standard for registry and discovery services.
Why a standard interface?

$n^*m$ hook-ups

$n+m$ hook-ups
Solver support

• All versions of OS download with COIN-OR solvers
  – Clp
  – Cbc
  – Ipopt
  – Bonmin
  – Couenne
  – Symphony

• Additional support
  – Cplex
  – GLPK
  – Lindo
OSSolverService capabilities

• OSSolverService can be run
  – locally or remotely
  – synchronously or asynchronously
  – with data local or remote relative to solver machine
  – as standalone application
  – from AMPL and GAMS
Running OSSolverService locally

- OSSolverService -config \
  ../data/configFiles/testlocal.config

- **testlocal.config** contains:
  - osil ../data/osilfiles/parincLinear.osil
  - osol ../data/osolfiles/parincLinear_ipopt.osol
  - solver ipopt
  - serviceMethod solve

- It is assumed that input files exist on the local host
OSSolverService on a remote server

- OSSolverService -config \
  ../data/configFiles/testremote.config

- testremote.config contains:
  - osil ../data/osilfiles/parincLinear.osil
  - osol ../data/osolfiles/parincLinear_ipopt.osol
  - solver ipopt
  - serviceMethod send
  - serviceLocation <url>

- It is assumed that input files exist on the remote server — otherwise they need to be uploaded first
Using OSAmplClient

Start `ampl.exe` at the command line. Inside `ampl.exe`, do the following

```plaintext
# open the AMPL model file
model hs71.mod;

# tell AMPL to use OSAmplClient as the solver
option solver OSAmplClient;

# now tell OSAmplClient to use Ipopt
option OSAmplClient_options "solver ipopt";

# tell ipopt to use a remote server (optional)
option ipopt_options "service http://gsbkip.uchicago.edu/os/OSSolverService.jws";

# solve the problem
solve;

# display the solution
display {j in 1.._nvars} (_varname[j], _var[j]);
```
GAMSlinks

- Implemented as a separate COIN-OR project
  
gams trnsport lp=os optfile=1
- This tells GAMS to read os.opt for more information
- os.opt looks like this
  
writeosil osil.xml
writeosrl osrl.xml
service
http://gsbkip.uchicago.edu/os/OSSolverService.jws
solver clp
OSiL

- XML schema for mathematical programs
  - Linear
  - Integer
  - Nonlinear
  - Stochastic
  - Multiobjective
  - Semidefinite
  - ...
OSiL Schema – Header information
Header information – Example

<?xml version="1.0" encoding="UTF8"?>
<osil xmlns="os.optimizationservices.org"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="OSiL.xsd">
  <instanceHeader>
    <name>FinPlan</name>
    <source>
      Birge and Louveaux, Stochastic Programming
    </source>
    <description>
      Three-stage stochastic investment problem
    </description>
  </instanceHeader>
  <instanceData>
    ...
  </instanceData>
</osil>
OSiL Schema – Deterministic data
OSInstance: In-memory representation

- XML elements correspond to C++ classes
- Child elements mapped as member classes

```cpp
class OSInstance{
public:
    OSInstance();
    InstanceHeader *instanceHeader;
    InstanceData *instanceData;
}; // class OSInstance
```

- `set()`, `get()` and `calculate()` methods
Instance vs. options

• Instance describes what is to be solved
  – Variables, objectives, relationships

• Options explain how to solve it
  – Algorithm tuning
    • e.g., tolerances, pricing and branching rules, starting point
  – Job performance
    • e.g., iteration limits, CPU limits
  – System requirements
  – Other, e.g., control of output levels
Solver option characteristics

• Different classes of options
• Many options shared among solvers
• Some options unique to one solver
• Syntax and meaning may vary
OSoL – OS option language

- Common syntax
- Solver-specific semantics
- Standard representation for common options
- Flexibility to allow extensions
- Solver driver translates options into form understandable by the solver
- In-memory representation: OSOption
- API: get(), set(), add() methods

© 2010 H.I. Gassmann
OSoL schema elements
OSoL optimization schema element
Sample .osol file

```xml
<?xml version="1.0" encoding="UTF-8"?>
<osol xmlns="os.optimizationservices.org"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="os.optimizationservices.org
http://www.optimizationservices.org/schemas/2.0/OSoL.xsd">
  <optimization numberOfVariables="2">
    <variables>
      <initialVariableValues numberOfVar="2">
        <var idx="0" value="5.0"/>
        <var idx="1" value="5.0"/>
      </initialVariableValues>
    </variables>
    <solverOptions numberOfSolverOptions="5">
      <solverOption name="tol" solver="ipopt" type="numeric" value="1.e-9"/>
      <solverOption name="print_level" solver="ipopt"
                    type="integer" value="5"/>
      <solverOption name="max_iter" solver="ipopt" type="integer"
                    value="2000"/>
      <solverOption name="LSgetParam_LP_PRINTLEVEL" solver="lindo"
                    category="model" type="integer" value="0"/>
      <solverOption name="LSgetParam_LP_PRINTLEVEL" solver="lindo"
                    category="environment" type="integer" value="1"/>
    </solverOptions>
  </optimization>
</osol>
```
OSrL and OSResult

• Result of the optimization
  – Solution status
  – Statistics
  – Value of primal and dual variables
  – Basis information

• Can be displayed in a browser

• In-memory representation: OSResult

• API: get(), set(), add() methods
Other recent developments

• Interactive shell
• Semidefinite programming
• Dip solver (decomposition for IP)
• Quadratic objectives for Clp and Cbc
How to get OS

• Download
  – Binaries
    • http://www.coin-or.org/download/binary/OS
      – OS-2.2.0-linux-x86_64-gcc4.3.2.tgz
  – Stable source
    • http://www.coin-or.org/download/source/OS/
      – OS-2.2.0.tgz
      – OS-2.2.0.zip
  – Development version (using svn)
    • svn co https://projects.coin-or.org/svn/OS/releases/2.2.0 COIN-OS
    • svn co https://projects.coin-or.org/svn/OS/trunk COIN-OS
QUESTIONS?

http://myweb.dal.ca/gassmann
http://www.optimizationservices.org
http://www.coin-or.org/projects/OS.xml

Horand.Gassmann@dal.ca

© 2010 H.I. Gassmann