Using COIN-OR Solvers with Visual Studio

Horand Gassmann, Jun Ma, and Kipp Martin

June 25, 2009

Abstract

This is for Windows users that want to run COIN-OR solvers directly from a file or a modeling language, or want to write applications with Visual Studio projects that use COIN-OR solvers. This download is plug-and-play it is not necessary to build any COIN-OR projects from source code.

Contents

1 The Download **3**

2 Calling COIN-OR Solvers with Model Instances **3**

3 Calling COIN-OR Solvers using a Modeling Language **6**

4 Using Visual Studio to Build Application **6**

5 Example Projects **6**

1 The Download

2 Calling COIN-OR Solvers with Model Instances

In the `bin` directory, the user will find the following solvers (see the project link for more information on the solvers):

- **bonmin.exe** – a solver for mixed-integer nonlinear programs– see `https://projects.coin-or.org/Bonmin`
- **cbc.exe** – a solver for mixed-integer linear programs – see `https://projects.coin-or.org/Cbc`
- **clp.exe** – a solver for linear programs – see `https://projects.coin-or.org/Clp`
- **couenne.exe** – a global optimizer for mixed-integer nonlinear programs – see `https://projects.coin-or.org/Couenne`
- **ipopt.exe** – an optimizer for continuous nonlinear programs – see `https://projects.coin-or.org/Ipopt`
- **symphony.exe** – a solver for mixed-integer linear programs – see `https://projects.coin-or.org/SYMPHONY`

See the project pages for a more detail on each of the solvers and which optimization instance format they take.

For the convenience of the user, the bin directory also contains the **OSSolverService.exe**. This executable is linked to libraries for all of the above solvers, and can be used in lieu of any of them. One advantage of using the *OSSolverService.exe* is its flexibility. You can call any of the above solvers with an instance in MPS, nl, or OSiL format. In addition, the **OSSolverService.exe** returns the solver solution in the OSrL XML format which is easily parsed.

Solve a linear program: using the `OSSolverService` . At the command line, connect (**cd**) to the bin directory and execute the following:

To solve a problem in OSiL XML format

```
OSSolverService -osil ../../data/osilFiles/parincLinear.osil
```

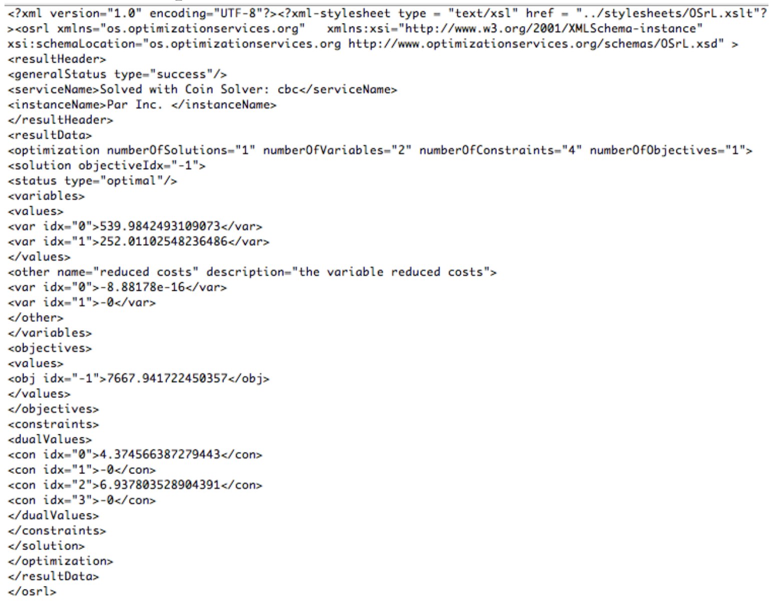
To solve a problem in AMPL nl format

```
OSSolverService -nl ../../data/amplFiles/parinc.nl
```

To solve a problem in MPS format

```
OSSolverService -mps ../../data/mpsFiles/parinc.mps
```

The result is printed in XML format:



```
<?xml version="1.0" encoding="UTF-8"?><?xml-stylesheet type = "text/xsl" href = "../stylesheets/OSrL.xslt"?
><osrl xmlns="os.optimizationservices.org"   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="os.optimizationservices.org http://www.optimizationservices.org/schemas/OSrL.xsd" >
<resultHeader>
<generalStatus type="success"/>
<serviceName>Solved with Coin Solver: cbc</serviceName>
<instanceName>Par Inc. </instanceName>
</resultHeader>
<resultData>
<optimization numberOfSolutions="1" numberOfVariables="2" numberOfConstraints="4" numberOfObjectives="1">
<solution objectiveIdx="-1">
<status type="optimal"/>
<variables>
<values>
<var idx="0">539.9842493109073</var>
<var idx="1">252.01102548236486</var>
</values>
<other name="reduced costs" description="the variable reduced costs">
<var idx="0">-8.88178e-16</var>
<var idx="1">-0</var>
</other>
</variables>
<objectives>
<values>
<obj idx="-1">7667.941722450357</obj>
</values>
</objectives>
<constraints>
<dualValues>
<con idx="0">4.374566387279443</con>
<con idx="1">-0</con>
<con idx="2">6.937803528904391</con>
<con idx="3">-0</con>
</dualValues>
</constraints>
</solution>
</optimization>
</resultData>
</osrl>
```

More detail – variables values

```
<values>
    <var idx="0">539.9842493109073</var>
    <var idx="1">252.01102548236486</var>
</values>
```

The objective function value

```
<objectives>
<values>
<obj idx="-1">7667.941722450357</obj>
</values>
</objectives>
```

You can also print the result to a file.
Use the osrl option

```
OSSolverService -osil ../../data/osilFiles/parincLinear.osil
    -osrl result.xml
```

You can display the result in a browser using XSLT.
Copy **data/stylesheets** into the root of the CoinAll distribution.
Open in your browser
To solve a **linear program** set the solver options to:

- clp
- dylp

To solve a **mixed-integer linear program** set the solver options to:

- cbc
- symphony

To solve a **continuous nonlinear program** set the solver options to:

- ipopt

To solve a **mixed-integer nonlinear program** set the solver options to:

- bonmin

Solving a **linear integer** program:

```
OSSolverService -osil ../../data/osilFiles/p0033.osil
      -solver cbc
```

Solving a **nonlinear** optimization problem

```
OSSolverService -osil ../../data/osilFiles/rosenbrockmod.osil
      -solver ipopt
```

Solving a **mixed-integer nonlinear** optimization problem

```
OSSolverService -osil ../../data/osilFiles/bonminEx1.osil
      -solver bonmin
```

It is possible to build the OSSolverService to work with other solvers but they are not included due to licensing issues.

- Glpk
- Cplex
- LINDO

Continually writing out command line options a pain. Use a `configuration` file instead. Do something like:

OSSolverService -config ../../data/configFiles/testLocal.config

where the file `testLocal.config` is

```
-osil ../../data/osilFiles/parincLinear.osil
-solver cbc
```

Note: the only option that is required is the location of an instance file

Finally – you can call solvers remotely.

Specify a **service location** of the remote solver service.

```
OSSolverService -osil ../data/osilFiles/parincLinear.osil
  -serviceLocation
     http://gsbkip.chicagogsb.edu/os/OSSolverService.jws
```

To get help

```
OSSolverService -h
```

–OR–

```
OSSolverService --help
```

See also for calling remote solvers.

3 Calling COIN-OR Solvers using a Modeling Language

Mention GAMSlinks

4 Using Visual Studio to Build Application

5 Example Projects