

Using the COIN-OR Server

Your CoinEasy Team

November 4, 2009

1 Overview

This document is part of the **CoinEasy** project. See projects.coin-or.org/CoinEasy. In this document we describe the options available to users of COIN-OR who are interested in solving optimization problems but do not wish to compile source code and build any of the COIN-OR projects. In particular, we show how the user can send optimization problems to a COIN-OR server and get results back. The COIN-OR server, webdss.ise.ufl.edu, is 2x Intel(R) Xeon(TM) CPU 3.06GHz 512MiB L2 1024MiB L3, 2GiB DRAM, 4x73GiB scsi disk 2xGigE machine. It allows the user to access the following COIN-OR optimization solvers:

- **Bonmin** – a solver for mixed-integer nonlinear optimization
- **Cbc** – a solver for mixed-integer linear programs
- **Clp** – a linear programming solver
- **Couenne** – a solver for mixed-integer nonlinear optimization problems and is capable of global optimization
- **DyLP** – a linear programming solver
- **Ipopt** – an interior point nonlinear optimization solver
- **SYMPHONY** – mixed integer linear solver that can be executed in either parallel (distributed or shared memory) or sequential modes
- **Vol** – a linear programming solver

All of these solvers on the COIN-OR server may be accessed through either the GAMS or AMPL modeling languages. In Section 2.1 we describe how to use the solvers using the GAMS modeling language. In Section 2.2 we describe how to call the solvers using the AMPL modeling language. In Section 3 we describe how to call the solvers using **OSSolverService**. The **OSSolverService** can be used independently of a modeling language. It can send optimization instances to the solvers in MPS, OSiL (a new XML based representation standard), AMPL nl, and GAM dat formats. A nice feature of the OSSolverService is that it can be used in asynchronous mode for large problems. This is described in Section 4.1. We show how to obtain a job id from the server, send a job to the server, check the server to see if the job is done, retrieve the job if it is done, and kill the job if it is taking too long. For sophisticated users we describe how to actually build applications that use the COIN-OR server in Section 4.2. Finally we describe how to download the necessary client software in Section 5. This software consists of executable programs for various platforms, the user is not required to compile code.

2 Calling the COIN-OR Server using a Modeling Language

2.1 Using GAMS

This section is based on the assumption that the user has installed GAMS (22.9 or above) on his or her machine. In the the distribution described in Section 5 there is a file `gmsos.zip`. Copy this file into your GAMS folder. Then run `gamsinst` and select `OS` as the default solver whenever it is listed as a solver. You can now solve a wide variety of problems either locally or remotely

through Optimization Services. In the discussion that follows we assume that folder where GAMS is installed is in the PATH command.

The distribution described in Section 5 contains a test problem, `p0033.gams`, which is a linear integer program. To solve this problem type

```
gams p0033.gams
```

This results in the test problem being solved on the local machine using the COIN-OR solver Cbc. In order to invoke the COIN-OR solver server it is necessary to provide GAMS with an option file. A key point to understand is that *in order to pass options to the solver on the server, the GAMS option file must contain the name of a solver option file that is passed to the server*. For example, in order to pass options to the solvers on the COIN-OR solver server create a GAMS option file `os.opt`. This option file should contain two options as follows:

```
service http://webdss.ise.ufl.edu:2646/OSServer/services/OSSolverService
readosol solveroptions.osol
```

Invoking the command

```
gams p0033.gams optfile 1
```

will result in GAMS generating a problem instance and sending to the server at the location `webdss.ise.ufl.edu`. GAMS will also pass to the remote server the options specified in the file `solveroptions.osol`.

The file `solveroptions.osol` is contained in the distribution described in Section 5. This file contains three solver options; two for Cbc, and one for SYMPHONY. In addition there is an XML tag `<solverToInvoke>`. This tag is used to tell the remote solver service which solver to invoke. The remote solver service will invoke Cbc by default. Hence it is important to set the value of this tag if a solve other than Cbc is desired.

It is also possible to generate the instance OSiL file, and write a solution OSrL file. If the file `os.opt` contained the following four options then

```
writeosil osil.xml
writeosrl osrl.xml
service http://webdss.ise.ufl.edu:2646/OSServer/services/OSSolverService
readosol solveroptions.osol
```

the model instance would be written to `osil.xml`, the solution result would be written to `osrl.xml`, and the option file `solveroptions.osol` would be sent to the server at `webdss.ise.ufl.edu`.

It is possible to define multiple GAMS options files following the specific naming conventions as set out below:

```
optfile=1 corresponds to <solver>.opt
optfile=2 corresponds to <solver>.op2
...
optfile=99 corresponds to <solver>.op99
```

For example, in order to solve the `p0033.gams` test problem using the options file `os.opt` execute the command

```
gams rbrock nlp os optfile=1
```

IMPORTANT: unlike the **OSAmplClient** described in Section 2.2 the GAMS add-on does not parse the **solveroptions.osol** file, it simply passes it onto the solver either locally or on the server.

GAMS SUMMARY:

1. Put **gmsos..zip** in the GAMS directory, execute **gamsinst** and select **OS** as the default solver whenever it is listed as a solver.
2. If no options are given, then the model will be solved locally and Clp will be used for linear programs, Cbc for integer linear programs, Ipopt for continuous nonlinear programs, and Bonmin for nonlinear integer.
3. In order to control behavior (for example, whether a local or remote solver is used) an options an option file, **os.opt**, must be used as follows

```
gams p0033.gms optfile=1
```

4. The **os.opt** file is used to specify *five potential options*:

- **service:** using the COIN-OR solver server, this is done by giving the option

```
service http://webdss.ise.ufl.edu:2646/OSServer/services/OSSolverService
```

- **readosol:** whether or not to send the solver an option file, this is done by giving the option

```
readosol solveroptions.osol
```

- **solver:** if a local solve is being done, a specific solver is specified by the option

```
solver solver_name
```

When the COIN-OR solver service is being used, the only way to specify the solver to use is through the `<solverToInvoke>` tag.

- **writeosrl:** the solution result can be put into an XML file by specifying the option

```
writeosrl osrl_file_name
```

- **writeosil:** the optimization instance can be put into an XML file by specifying the option

```
writeosil osil_file_name
```

2.2 Using AMPL

This section is based on the assumption that the user has installed AMPL on his or her machine. It is possible to call all of the COIN-OR solvers listed in Section 1 directly from the AMPL (see <http://www.ampl.com>) modeling language. In this discussion we assume the user has already obtained and installed AMPL. In the download described in Section 5 there is an executable, **OSAmplClient.exe** that is linked to all of the COIN-OR solvers listed in Section 1. From the perspective of AMPL, the **OSAmplClient** acts like an AMPL “solver”. The **OSAmplClient** can be used to solve problems either locally or remotely. In the following discussion we assume that the

AMPL executable `ampl.exe`, the `OSAmplClient`, and the test problem `parIncEx.mod` are all in the same directory.

The problem instance `parIncEx.mod` is an AMPL model file included in distribution 5. To solve this problem locally by calling the `OSAmplClient` from AMPL first start AMPL and then open the `parIncEx.mod` file inside AMPL.

```
# take in sample LP problem
# assume the problem is in the AMPL directory
model parIncEx.mod;
```

The next step is to tell AMPL that the solver it is going to use is `OSAmplClient`. Do this by issuing the following command inside AMPL.

```
# tell AMPL that the solver is OSAmplClient
option solver OSAmplClient;
```

It is not necessary to provide the `OSAmplClient` solver with any options. You can just issue the `solve` command in AMPL as illustrated below.

```
# solve the problem
solve;
```

If no options are specified, then the default solver `Cbc` is invoked and the problem is solved on the local (client) machine. Next, assume that you have a large problem you want solve on the remote solver. It is necessary to specify the location of the server solver as an option to `OSAmplClient`. This is done as follows.

```
# now tell OSAmplClient to use the remote solver
option OSAmplClient_options "serviceURI http://webdss.ise.ufl.edu:2646/OSServer/services/OSSolverService";
```

This will send the problem to the solver server at location **webdss.ise.ufl.edu**. It is also easy to specify a solver other than the default solver. For example, in order to specify the `SYMPHONY` solver use the `solver` option of `OSAmplClient`. Invoking `SYMPHONY` on the remote server is done as follows.

```
# now tell OSAmplClient to use the remote solver
option OSAmplClient_options "solver symphony serviceURI http://webdss.ise.ufl.edu:2646/OSServer/services/OSSolverService";
```

Always specify the name of the solver entirely in lower case. Finally, the user may wish to pass options to a solver. This is done by providing an options file. A sample options file, **solveroptions.osol** is provided with this distribution. The sequence of options including the name of the options file is given below.

```
model parIncEx.mod;
option solver OSAmplClient;
option OSAmplClient_options "solver symphony optionFile solveroptions.osol serviceURI http://webdss.ise.ufl.edu:2646/OSServer/services/OSSolverService";
solve;
```

Different solvers have different options, and we recommend that the user look at the documentation for the solver of interest in order to see which options are available.

It is also possible to specify the name of the solver and the server location in the options file. Indeed, if you examine the file **solveroptions.osol** you will see that there is an XML tag `<solverToInvoke>` and that the solver given is **symphony**. There is also an XML tag `<serviceURI>` that can be used to specify the location of the server. For the option file **solveroptions.osol** executing

```
option OSAmplClient_options "solver symphony optionFile solveroptions.osol serviceURI http://webdss.ise.ufl.edu:2646/OSServer/services/OSSolverService";
```

has exactly the same effect as

```
option OSAmplClient_options "optionFile solveroptions.osol";
```

It is possible to override the solver option and the server location given in the option file. For example,

```
option OSAmplClient_options "optionFile solveroptions.osol solver bonmin";
```

will invoke the **bonmin** solver rather than symphony.

AMPL SUMMARY:

1. tell AMPL to use the OSAmplClient as the solver:

```
option solver OSAmplClient;
```

2. specify options to the OSAmplclient solver by using the AMPL command **OSAmplClient_options**
3. there are three possible options to specify:
 - the name of the solver using the **solver** option
 - the location of the remote server using the **serviceURI** option
 - the location of the option file using the **optionFile** option
4. the **solver** and **service URI** options can be specified in the options file using the `<solverToInvoke>` and `<serviceURI>` XML tags, respectively.
5. specifying the **solver** or **serviceURI** directly through **OSAmplClient_options** will override the settings in the options file
6. if no options are specified using **OSAmplClient_options**, then the **Cbc** solver will be invoked locally by **OSAmplClient**
7. the options given to **OSAmplClient_options** can be given in any order

3 Calling COIN-OR Solvers Using OSSolverService

Make sure to illustrate using an MPS file and an nl file.

4 More Sophisticated Methods

Explain that using the modeling languages involve a synchronous call. Explain that we might want to submit jobs asynchronously. In doing so we need a job id, need to be knock the server to see if the job is done, and then retrieve the results.

4.1 Making Asynchronous Calls

4.2 Writing Applications to Call the COIN-OR Server

5 Get the Client Software

Describe the binary and how to obtain.

This distribution contains:

- OSAmplClient
- OSSolverService
- gmsos_.zip (to be put in the GAMS directory if you use GAMS)
- parIncEx.mod an AMPL test problem
- p0033.gms a GAMS test problem
- solveroptions.osol – a sample options file