

DOCUMENTOS DE TRABAJO

BILTOKI

D.T. 2011.08

On Downloading and Using CPLEX within COIN-OR
for Solving Linear/Integer Optimization Problems.

Gloria Pérez y M. Araceli Garín

eman ta zabal zazu



Universidad Euskal Herriko
del País Vasco Unibertsitatea

Facultad de Ciencias Económicas.
Avda. Lehendakari Aguirre, 83
48015 BILBAO.

Documento de Trabajo BILTOKI DT2011.08

Editado por el Departamento de Economía Aplicada III (Econometría y Estadística)
de la Universidad del País Vasco.

ISSN: 1134-8984

On downloading and using CPLEX within COIN-OR for solving linear/integer optimization problems

Gloria Pérez¹ and M. Araceli Garín²

¹ Dpto. Matemática Aplicada y Estadística e I.O. Universidad del País Vasco, Leioa (Vizcaya), Spain.

email: gloria.perez@ehu.es

² Dpto. Economía Aplicada III. Universidad del País Vasco, Bilbao (Vizcaya), Spain.

email: mariaaraceli.garin@ehu.es

November, 2011

Abstract

The aim of this technical report is to present some detailed explanations in order to use the solver CPLEX within COIN-OR environment. In particular, we describe how to download, install and use the corresponding source code and libraries under Windows and Linux operating systems. We will use an example taken from the literature, with the experimental code and files written in C++, to describe the whole process of editing, compiling and running the executable, to solve this optimization problem by using this software. In the case of the Windows environment, a C++ compiler is also needed. We will use the Visual C++ 2010 Express Edition.

Keywords: CPLEX, COIN-OR, C++

1. Introduction

COIN-OR, which stands for Computational Infrastructure for Operations Research, is a collection of open source software for optimization, see <http://www.coin-or.org>. The Open Source Initiative (OSI) is a non-profit corporation with global scope formed to build bridges among different constituencies in the open source community. There exist several solvers with OSI interfaces in COIN-OR environment. One of them is IBM ILOG CPLEX, see <http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/>. In this case, the open access to this software will be done with an academic license. The source code in case of COIN-OR and libraries in case of CPLEX will be compiled and linked with your own code.

In this working paper we describe how to download, install and use the corresponding solvers under Windows and Linux operating systems. In both cases we present the installation options for 32 bits machines, although it is possible and very similar the installation in 64 bits machines. We will use an example with a principal program written in C++, for solving a small optimization problem. Additionally to this principal program, we will compile several auxiliary functions, written as external files (also with extension .cpp) and describe how to link this code with COIN and CPLEX libraries and with some others libraries with general and interface applications to generate the executable for solving mixed integer optimization problems.

In order to illustrate the procedure step by step, we will use the farmer's problem taken from Birge and Louveaux (1997). We will describe the whole procedure under Windows and Linux-like operating systems. In Pérez and Garín (2010), these same authors present a similar description for using COIN-OR software in the solution of linear/integer optimization problems. The remainder of the paper is as follows: Section 2 presents an example taken from the literature, over we have written the source code. Section 3 describes the downloading process and basic installations of CPLEX, COIN-OR and Visual C++ 2010 under Windows-like systems. Section 4 presents the details to create the Visual C++ project, in order to generate an executable (solution) with CPLEX within COIN-OR environment and running it, to solve an optimization problem. Section 5, gives the main steps for the basic installation of CPLEX software within COIN-OR under Linux-like systems. Section 6, describes the process to link your own code with CPLEX and COIN-OR environment and running the executable under Linux. Appendix A summarizes the source code files written in C++, for the example. Appendix B adds the Makefile file for compiling and linking under Linux, and Appendix C presents the output file to check the results of farmer problem. Finally, Appendix D presents an easy interactive way to solve the optimization problems with CPLEX. The working paper finishes with some references.

2. Illustrative case

Let us consider the farmer's problem taken from the Section Introduction and Examples, pp. 4-15, of Birge and Louveaux (1997). The problem reads as follows:

$$\begin{aligned}
\min \quad & 150x_1 + 230x_2 + 260x_3 - \frac{1}{3}(170w_{11} - 238y_{11} + 150w_{21} - 210y_{21} + 36w_{31} + 10w_{41}) \\
& - \frac{1}{3}(170w_{12} - 238y_{12} + 150w_{22} - 210y_{22} + 36w_{32} + 10w_{42}) \\
& - \frac{1}{3}(170w_{13} - 238y_{13} + 150w_{23} - 210y_{23} + 36w_{33} + 10w_{43}) \\
\text{s.t.} \quad & x_1 + x_2 + x_3 \leq 500, \\
& 3x_1 + y_{11} - w_{11} \geq 200, \\
& 2.5x_1 + y_{12} - w_{12} \geq 200, \\
& 2x_1 + y_{13} - w_{13} \geq 200, \\
& 3.6x_2 + y_{21} - w_{21} \geq 240, \\
& 3x_2 + y_{22} - w_{22} \geq 240, \\
& 2.4x_2 + y_{23} - w_{23} \geq 240, \\
& w_{31} + w_{41} \leq 24x_3, \\
& w_{32} + w_{42} \leq 20x_3, \\
& w_{33} + w_{43} \leq 16x_3, \\
& w_{31} \leq 6000, \\
& w_{32} \leq 6000, \\
& w_{33} \leq 6000, \\
& x_i, y_{j\omega}, w_{i\omega} \geq 0, \quad \forall i, j, \omega
\end{aligned}$$

where x_1, x_2 and x_3 are the first stage variables, i.e., represent decisions on land assignment, that have to be taken now. However, $w_{i\omega}$ $i = 1, \dots, 4$ and $y_{j\omega}$ $j = 1, 2$, are second stage variables, which represent decisions about sales and purchases, that depend on the yields. These decisions depend also of a scenario index ω , with $\omega = 1, 2, 3$, which corresponds to above average, average or below average yields, respectively. Assuming that the farmer wants to maximize long-run profit, it is reasonable for him a solution that maximizes his expected profit. If the three scenarios have an equal probability of $\frac{1}{3}$, the farmer' problem reads as given above. The optimal solution value is 108390, as you can check in the output file, *resul-farmer.dat*, given in the Appendix C.

At the beginning, we need to edit several files with dimensions, auxiliary functions and the principal program corresponding to the source code in order to define the problem statement, see Appendix A for the source files. We consider the two stage stochastic problem in compact representation,

$$\begin{aligned}
\min \quad & c_1^t x + \sum_{\omega \in \Omega} p_\omega c_{2\omega}^t y_\omega \\
\text{s.t.} \quad & \\
& b_1 \leq Ax \leq b_2 \\
& h_{1\omega} \leq T^\omega x + W^\omega y_\omega \leq h_{2\omega} \\
& x, y_\omega \geq 0
\end{aligned}$$

where vector x denotes the first stage variables and the vectors y_ω and w_ω the second stage variables.

To introduce the dimensions we must edit a file named *const-farmer.h*. We do not have the .mps file with the coefficients, so we are going to introduce them by indices into the arrays of COIN-OR. The coefficients of the model are charged with the auxiliary function named *model-farmer.cpp*. This information must be included in the arrays of COIN-OR. This is made by the auxiliary function *param-*

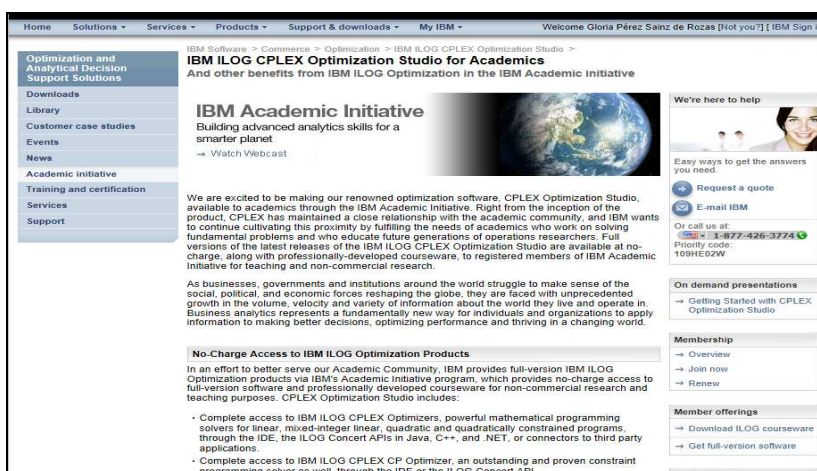
farmer.cpp. Finally, the principal program code is in file *principal-farmer-cplex.cpp*. In this program is also described how to solve the model if the information is read from a mps file. After solving the linear problem, we have added the possibility of solving a mixed-integer problem by considering for example, the first stage variables as integer. A header file includes the needed solvers of COIN-OR and CPLEX. This is the file *pm.h*.

3. The downloading process and basic installations of CPLEX, COIN-OR and Visual C++ 2010 under Windows-like systems

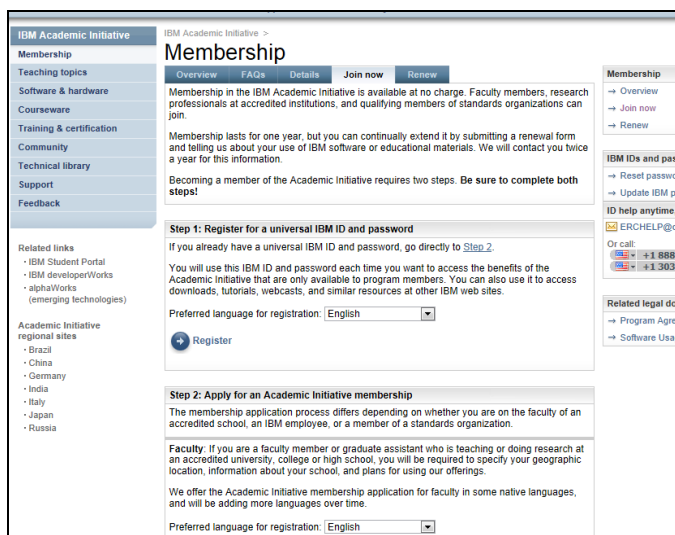
The first step is to download the COIN-OR code. For doing it, you must click on [Download/Use](#) in the left hand side of the home page <http://www.coin-or.org>. Then in the second Section titled Source Code, you must click on [here](#) to download the source code for the latest stable release. You can observe an index for the source of a number of COIN projects. You must click on [CoinAll/](#) to obtain the list of last versions. In this case you will click on and select [CoinAll-1.4.0.zip](#). Alternatively, you can go directly to the home page <http://www.coin-or.org/download/source/CoinAll/CoinAll-1.4.0.zip>. After saving the corresponding zip file, you must extract the code into the subdirectory named, C:\CoinAll\.

The second step is to download the CPLEX libraries with academic license. To do it, go to the page <http://www-01.ibm.com/software/websphere/products/optimization/academic-initiative/index.html>

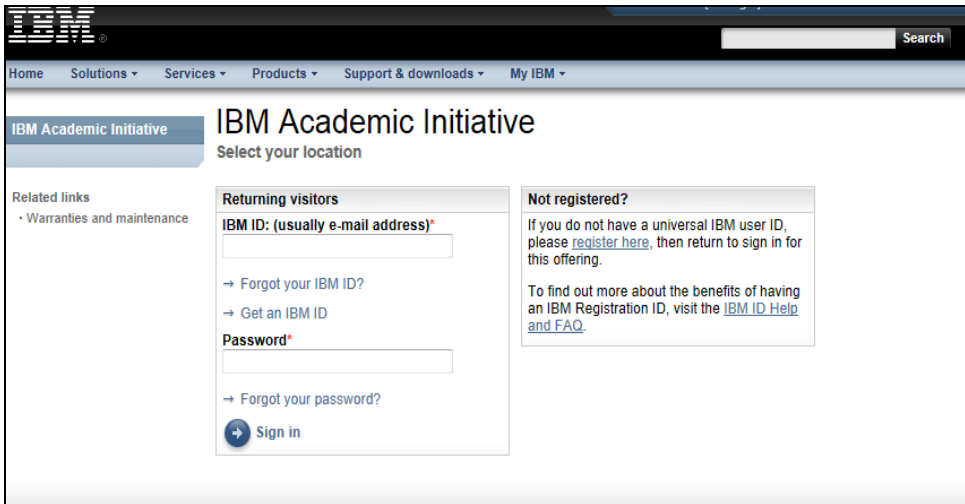
In the right hand side, middle page and under the epigraph Membership, you must click on [Join now](#), for registration.



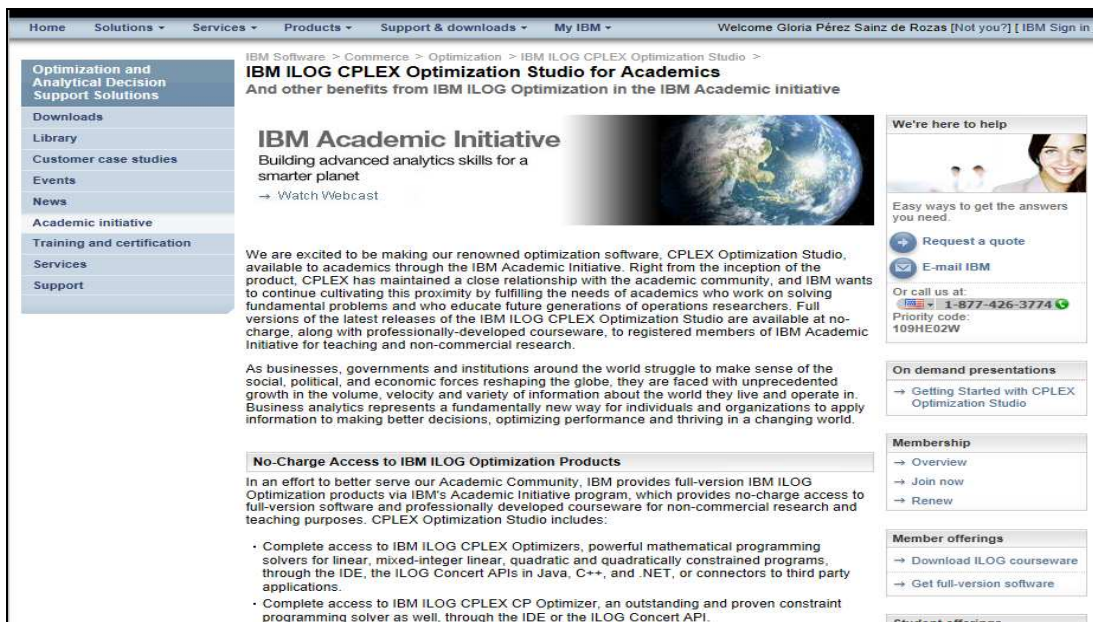
Becoming a member of the Academic Initiative require two steps. In the first step, click on [Register](#), and complete the required information.



You will see the following screen. Then you must give your institutional (UPV/EHU, if case) e-mail address, and a password, and click on [Sign in](#) button.



After obtaining your membership, you can access to the IBM Academic Initiative resources. You can back to the IBM Academic Initiative home page, an click on the second option [Get full-version software](#) of the epigraph Member offerings under the Membership epigraph.



After clicking on, you will see the following screen.

Get software & system access

Offered through the Academic Initiative program

[Download software](#)
[Request software CDs](#)
[Request virtual access](#)

Software for members (Eligible product list)

IBM Academic Initiative members can get full versions of a large selection of IBM software, at no charge. Most of the software can be downloaded from the Software Catalog, and a few additional products are available on CD by request. Members can also request virtual access to some software through the Amazon Web Services cloud or to IBM Power Systems or System z enterprise computing environments.

To download or request any IBM software, you must be a member of the IBM Academic Initiative. Not a member? [Join now!](#)

Download full-version software

All software downloads are available to members at no charge. The Software Catalog contains products from each IBM software brand: Information Management (including Cognos), Lotus, Rational, Tivoli, and WebSphere (including ILOG). You can use them to teach about database management, team collaboration, software development, systems performance and management, Web services, and many other technologies.

You can search for software in the catalog by specifying text (such as a product name), a category, or a part number.

- Download from the Software Catalog

Note about license keys: Most ILOG Optimization products that are available to Academic Initiative members will require a license key, and most Rational products will require either a license key or an activation kit. Be sure to read the [ILOG support](#) and [Rational support](#) pages for detailed instructions.

Request software on CDs

This is a summary of the products available on CD. The request form lists the specific products and versions.

- AIX operating system and documentation
- Cluster Systems Management for AIX and Linux
- Engineering and Scientific Subroutine Library (ESSL) for AIX and Linux
- Parallel ESSL and Parallel Environment for AIX and Linux
- Performance Toolbox and Performance AIDE for AIX
- TWS Loadleveler for AIX

Please note that each Academic Initiative member may request a maximum of one CD per product. We do not offer a CD that contains all of our products; you must request each product CD individually.

Membership

- Overview
- Join now
- Renew

PowerPC 405 Core offer

The specifications of the PowerPC 405 Core are freely available to Academic Initiative members.

- Find out more

Publicly available software

Download the developer or community edition of these products. Membership is not required.

- Apache Hadoop (IBM Distribution)
- DB2 Express-C
- DB2 Express-C Virtual Appliance
- Informix Virtual Appliance
- Lotus Symphony
- Rational EGL Community Edition
- WebSphere Application Server Community Edition
- WebSphere sMash Developer Edition

Select the [Download from the Software Catalog](#) option in the middle of the page. You will see the page entitled: IBM Academic Initiative Program, where you must click on [Submit](#) button if you are agree that IBM may process you data. After submitting, you go to the following page and in the left had side of the screen you must click on the first item [Search for software](#).

You will see the following screen. You must type CPLEX in the box of Find by search text. Before clicking on [Search](#), to filter your search, select in the icon [Search filter options](#), the Operating System *Windows XP*, and Language *English*. Moreover, in the box of Find by part number, you can type *CZJS1ML* and click on the search button.

The screenshot shows the IBM Academic Initiative Software Downloads page. The search bar contains the text "CPLEX" and a search button. Below the search bar, a dropdown menu displays search results for "CPLEX". The first result is "IBM ILOG CPLEX Optimization Studio Academic Research Edition V12.3 Multiplatform Multilingual eAssembly". The second result is "IBM ILOG CPLEX Optimization Studio Academic Research Edition V12.2 Multiplatform Multilingual eAssembly". The page also includes a navigation menu, a sidebar with "Search for software" and "Popular downloads", and a "Related links" section.

The selected version must be IBM ILOG CPLEX Optimization Studio Research Edition v12.2 for Windows 32 bits Multilingual. Also, you can sign the license agreement.

IBM Academic Initiative > Search for software >

Find by search text results

Find by search text criteria

Search text:

Search filter options

Download method: Download Director Http transfer

eAssemblies (2) Images (27)

Sort results by: Default

Find by search text results

Expand all Collapse all

Use check boxes to select image(s) to download.

WebSphere Software (2 eAssemblies : 24 Images)

<input checked="" type="checkbox"/> IBM ILOG CPLEX Optimization Studio Academic Research Edition V12.2 Multiplatform Multilingual eAssembly (CRC3IML)	Size	12 Images (2,278mb)
	Date posted	23-jul-2010
<input checked="" type="checkbox"/> IBM ILOG CPLEX Optimization Studio Academic Research Edition V12.3 Multiplatform Multilingual eAssembly (CRFA1ML)	Size	12 Images (3,246mb)
	Date posted	22-jul-2011

Click also on **I agree**, and at the end of the following screen, and then, click on the [Download now](#) button.

eAssemblies (1) Images (15)

Sort results by: Default

Find by search text results

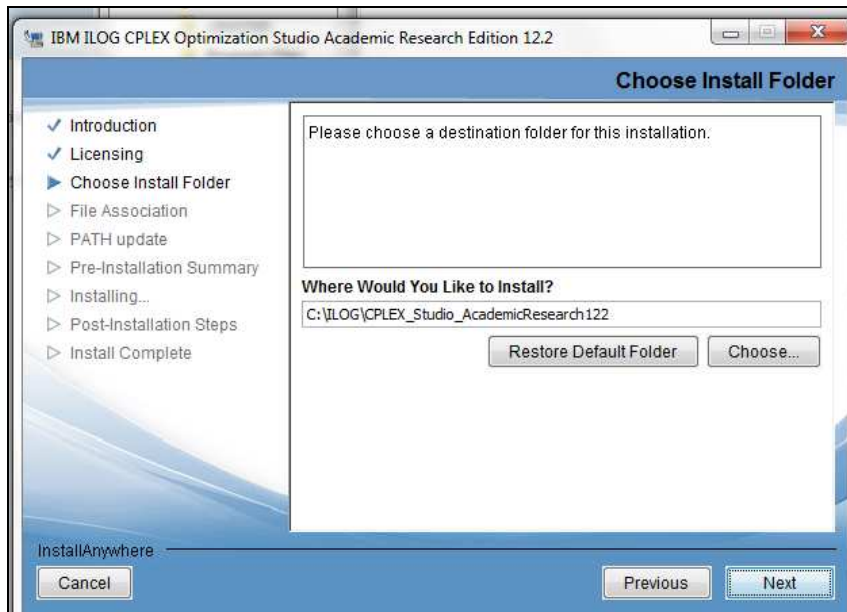
Expand all Collapse all

Use check boxes to select image(s) to download.

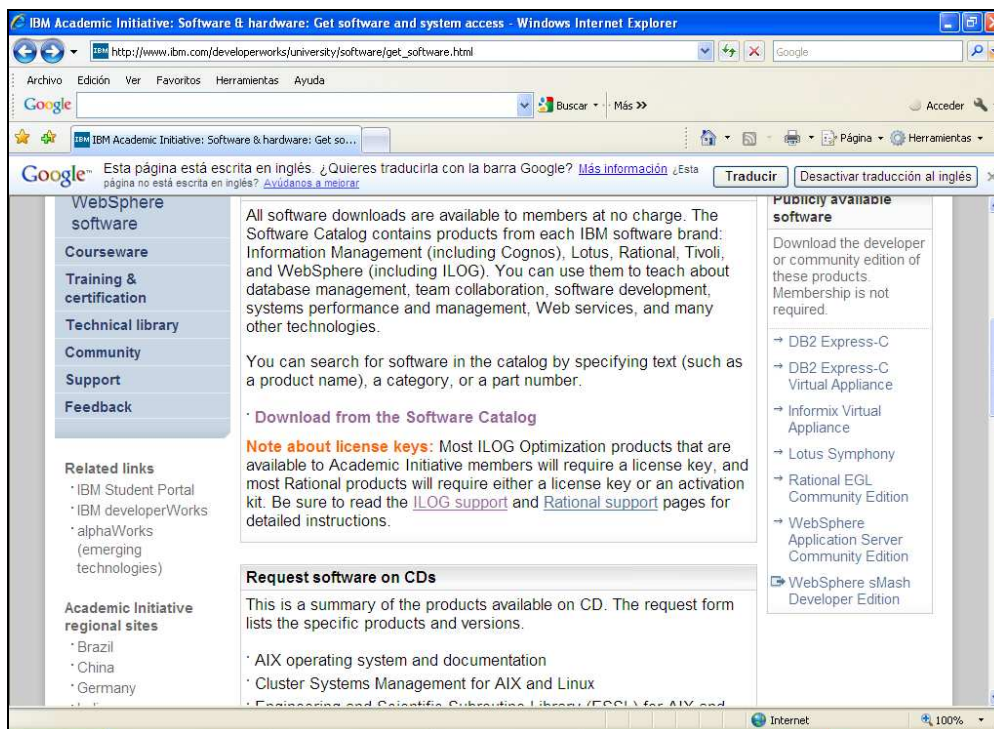
WebSphere Software (1 eAssembly : 12 Images)

<input checked="" type="checkbox"/> IBM ILOG CPLEX Optimization Studio Academic Research Edition V12.2 Multiplatform Multilingual eAssembly (CRC3IML)	Size	12 Images (2,278mb)
	Date posted	23-jul-2010
Multi-product package terms		
Important! This software is licensed as an eAssembly. You may not transfer or remarket individual products from it as this would violate the license terms applicable to the eAssembly. Your acceptance of the license terms applicable to the eAssembly is a precondition to your downloading and using the eAssembly.		
<input type="checkbox"/> Select All		
<input checked="" type="checkbox"/> IBM ILOG CPLEX Optimization Studio Academic Research Edition V12.2 for Windows 32 bits Multilingual (CZJS1ML) - View details	Size	294mb
	Date posted	23-jul-2010

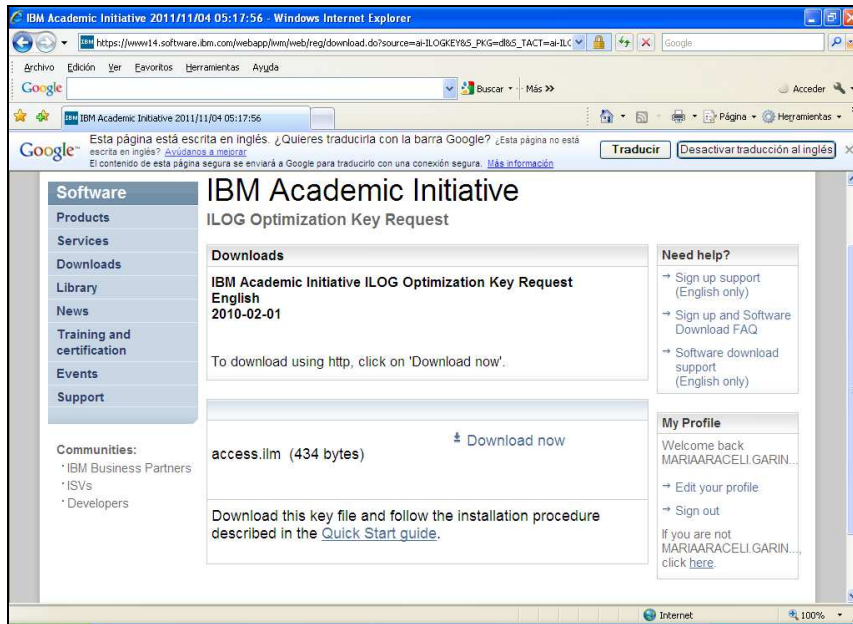
The executable file is called **cplex_studio122.acad.win-x86-32.exe**. Select the directory to install it, with name C:\ILOG\CPLEX_Studio_AcademicResearch122. Clicking two times over the .exe file, it will begin the process of installation, you must select the folder of the installation,



IBM ILOG CPLEX is a product that needs a license key. Go back to the home page from where you have downloaded the code, and click on **ILOG support**, in the paragraph **Note about license keys**.



You must follow step by step the process for obtaining the license key. Click on the **ILOG Optimization Key Request** button at Step 2. As membership, with your ID and password, you will obtain the following screen. Click on **Download now** button, and you will get a text plain file, the key file access.ilm. Follow the guide **ILOGQuickStart.pdf** for its installation. To obtain this file, please click on **Quick Start guide** button. Save a copy of the license key file **access.ilm** in the location C:\ILOG\ILM\.

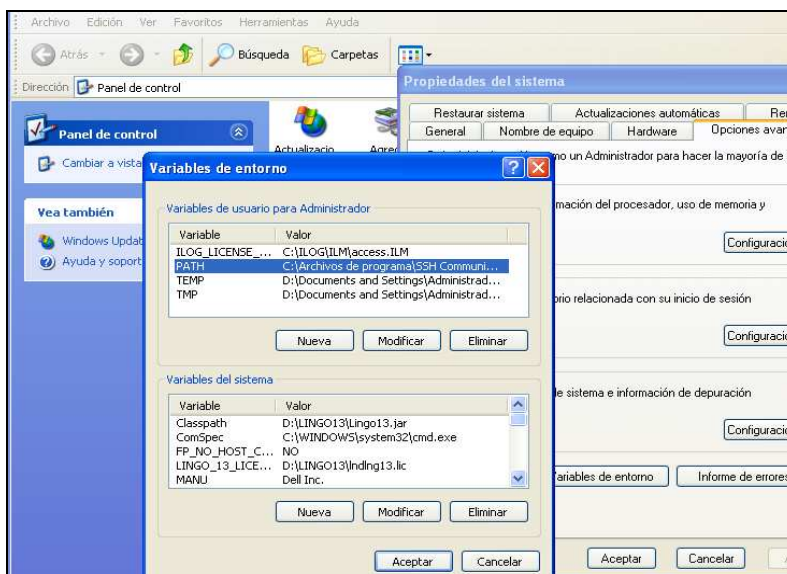


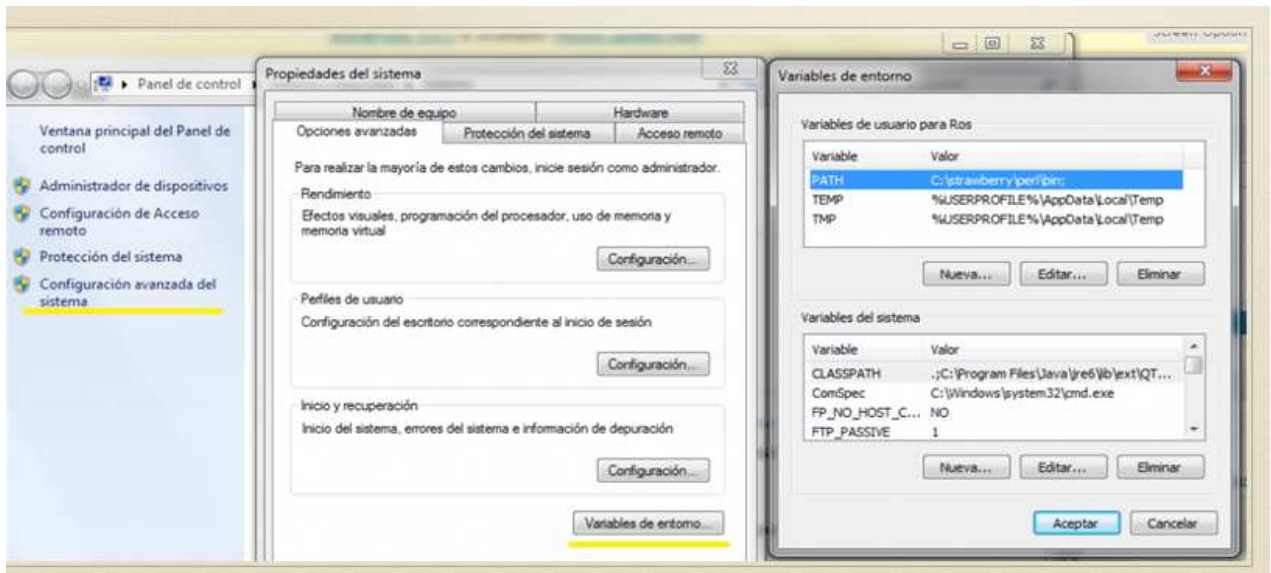
Once you have downloaded and installed the source code and libraries of COIN-OR and IBM ILOG CPLEX, you must do the following movement of files. Look at the file **cplex.h** in the following directory
 C:\ILOG\CPLEX_Studio_AcademicResearch122\cplex\include\ilcplex.

Copy the file **cplex.h** into the directory C:\CoinAll\Osi\src\OsiCpx.

Now, go to the directory C:\CoinAll\Osi\src, and locate the following six header files: **OsiSolverInterface.hpp**, **OsiCollections.hpp**, **OsiSolverParameters.hpp**, **OsiCut.hpp**, **OsiRowCut.hpp**, and **OsiColCut.hpp**. Copy all of them to the subdirectory C:\CoinAll\Osi\src\OsiCpx.

Finally, you must add two environment variables in your computer. To do it, go to your Desktop, and click on the Inicio button. Then, click on the buttons: **Panel de control -> Sistema -> Opciones Avanzadas -> Variables de entorno**. Under the epigraph Variables de usuario, you must add two new variables. Click on the Nueva button add a first variable with name PATH, and value, C:\ILOG\CPLEX_Studio_AcademicResearch122\cplex\bin\x86_win32. Then, add a second variable with name ILOG_LICENSE_FILE and value C:\ILOG\ILM\access.ilm. After doing it, you must restart your computer. The corresponding screens under Windows XP and Windows 7, respectively, are as follow:

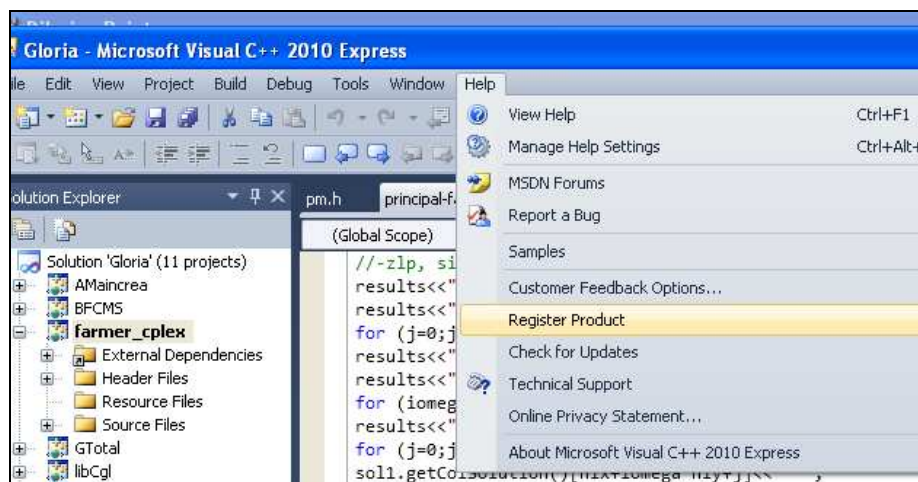


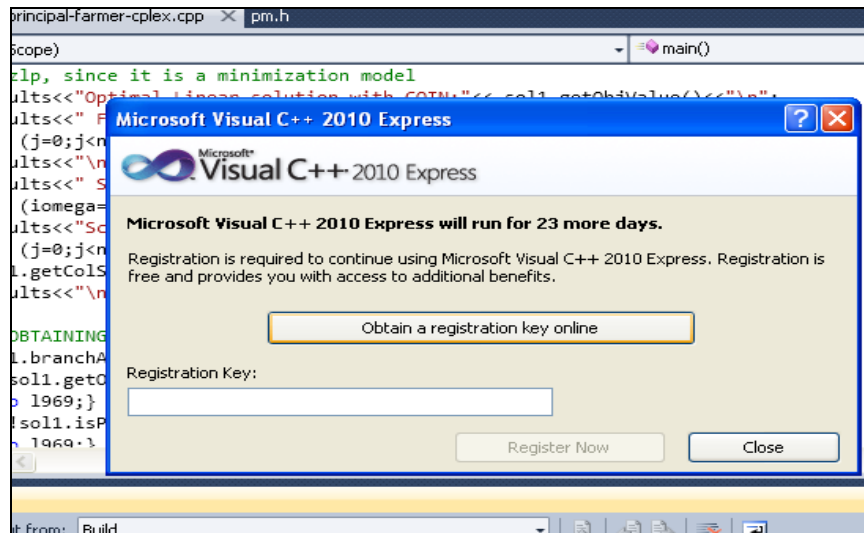


The third piece that you need for running your source code is a C++ compiler. You can download the Visual C++ 2010 Express Edition (30/90 days free) from the home page <http://www.microsoft.com/Express/VC>. Click on the Visual C++ 2010 express button in the right hand side of the screen, select the language and click on the INSTALAR AHORA button. After downloading and save the file vc_web.exe anywhere in your computer, you must click two times over it to execute it and start the installation procedure. When it finishes, you will see the following screen.



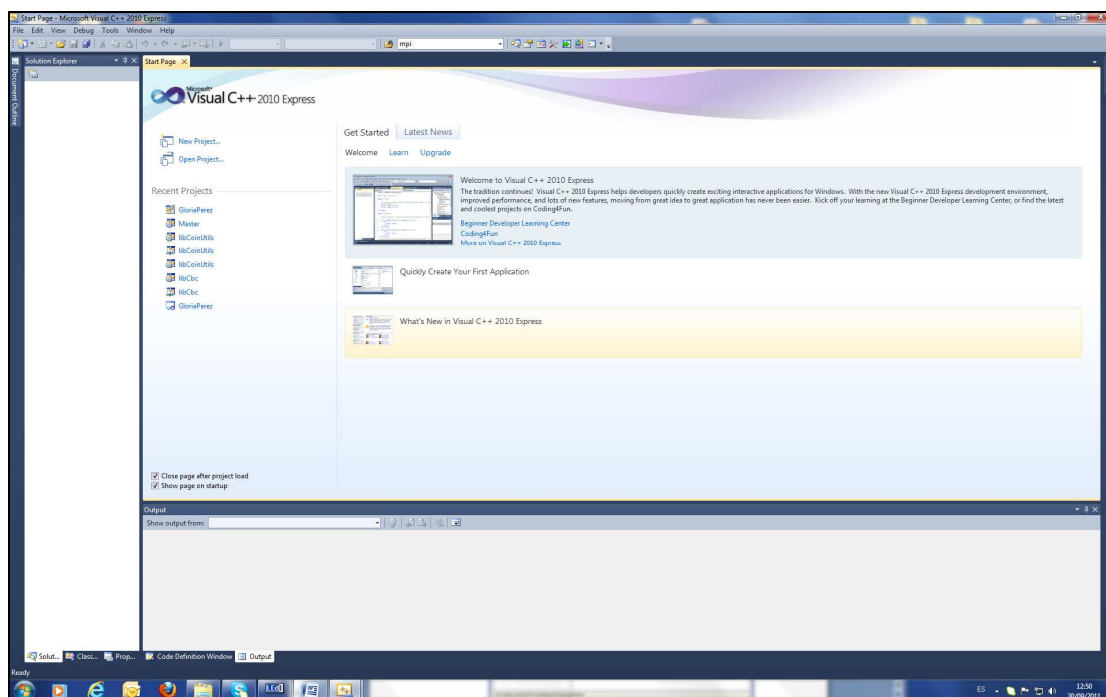
Now, you must obtain a license key for using the free version of the C++ compiler. To obtain it, go to the start page of Visual C++ 2011 and select the following sequence of buttons, Start page-> Help->Register Product->Obtain a registration key online.





You need a Hotmail account, and a password to get the fourteen digits code, that you have to copy in the registration key window and obtain the free use of the product (Visual C++ 2010).

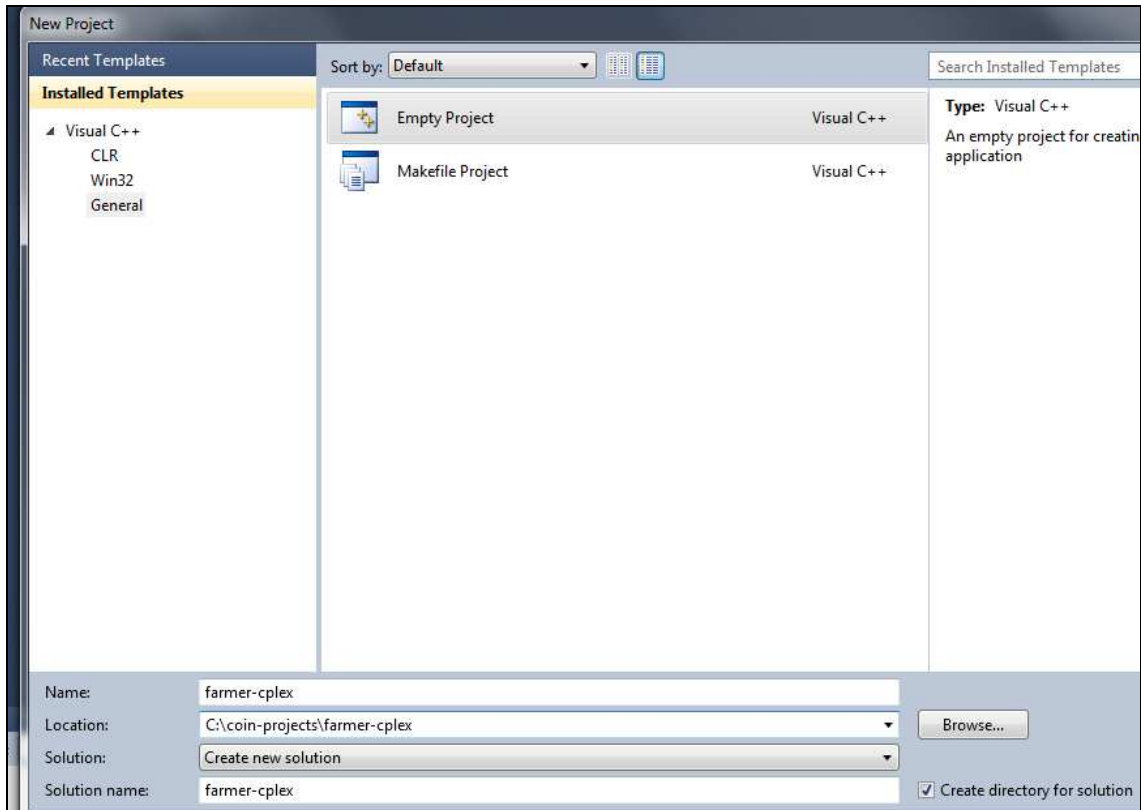
Now you can access to the compiler start page, that might look like this:



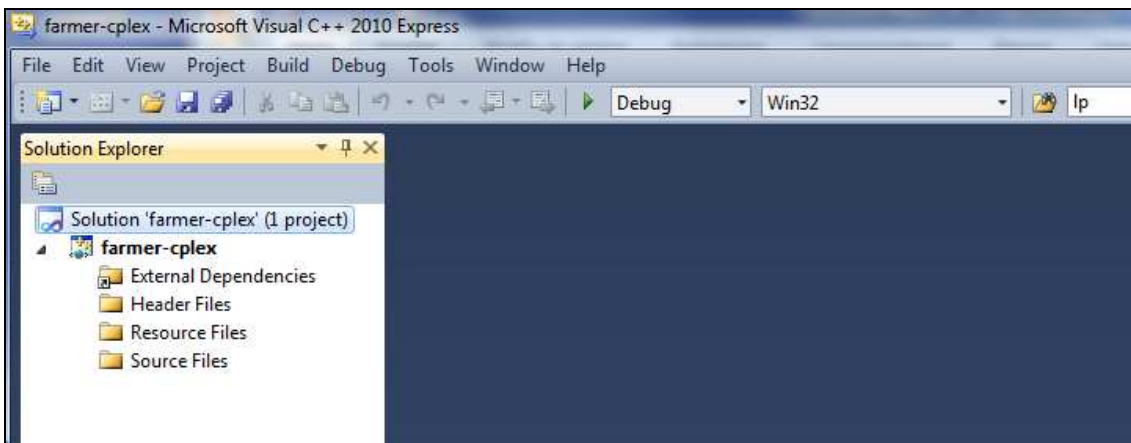
4. Linking your code with CPLEX and COIN-OR and running the executable under Windows

You must create a Visual C++ project to compile and link your code. Remember that you must establish a working directory, for example C:\coin-projects\farmer-cplex, where your source code is.

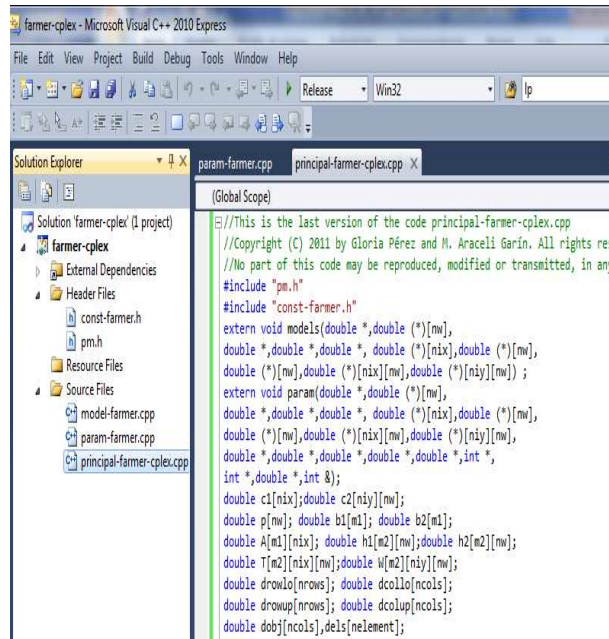
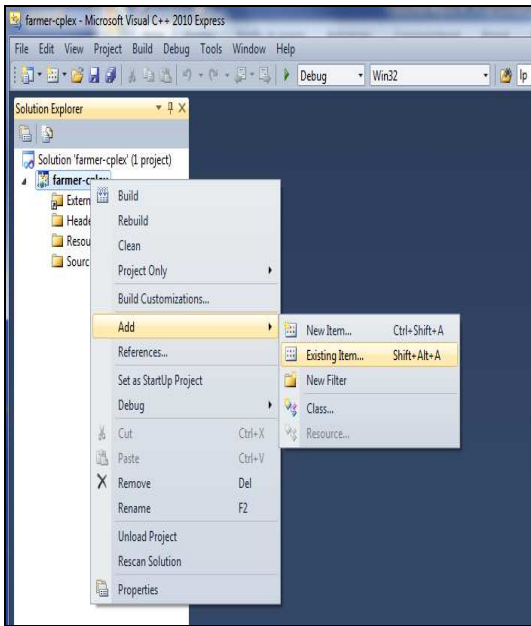
From the Desktop, click two times on Visual C++ 2010 to open it. You must select New Project in the left hand side options menu. Then select the buttons General->Empty Project, and you will get a screen like this:



You must choose a name for your project, for example “farmer-cplex”, and enter the working directory C:\coin-projects\farmer-cplex as location where you have to write your programs .cpp and .h, and a name if you want to create a new subdirectory for the solution. Also you can select to create the project in the current *solution* or to create a new *solution*. If you click on OK, the project *farmer-cplex* and the solution *farmer-cplex* are created, and look like this:



In the working directory you have your own code, the source code .cpp and header .h files shown in the Appendix A. You must click on File and open the file *principal-farmer-cplex.cpp* from the Visual C++ compiler. You can also do it, by adding your own source files of code (.cpp) and the headers (.h) as existing items. To do it, you must click on the right button over the Source Files button, and over the Header Files button. Then, you can open the files clicking two times over its name. The screen, look like this:



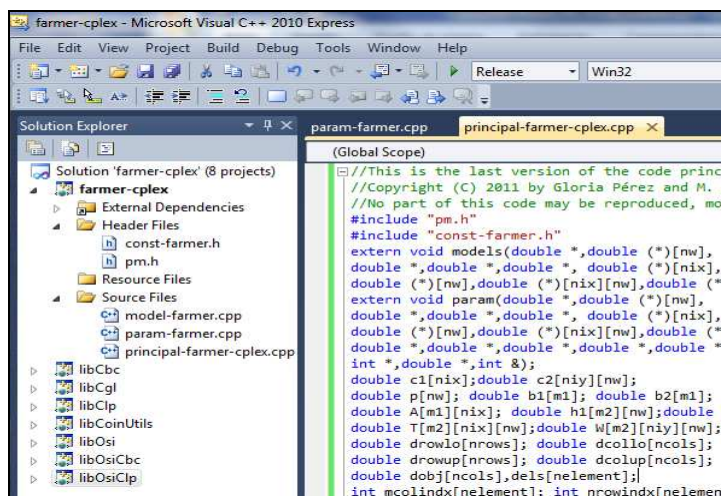
Also you must change the Solution Configurations in the front of the screen, from Debug to Release.



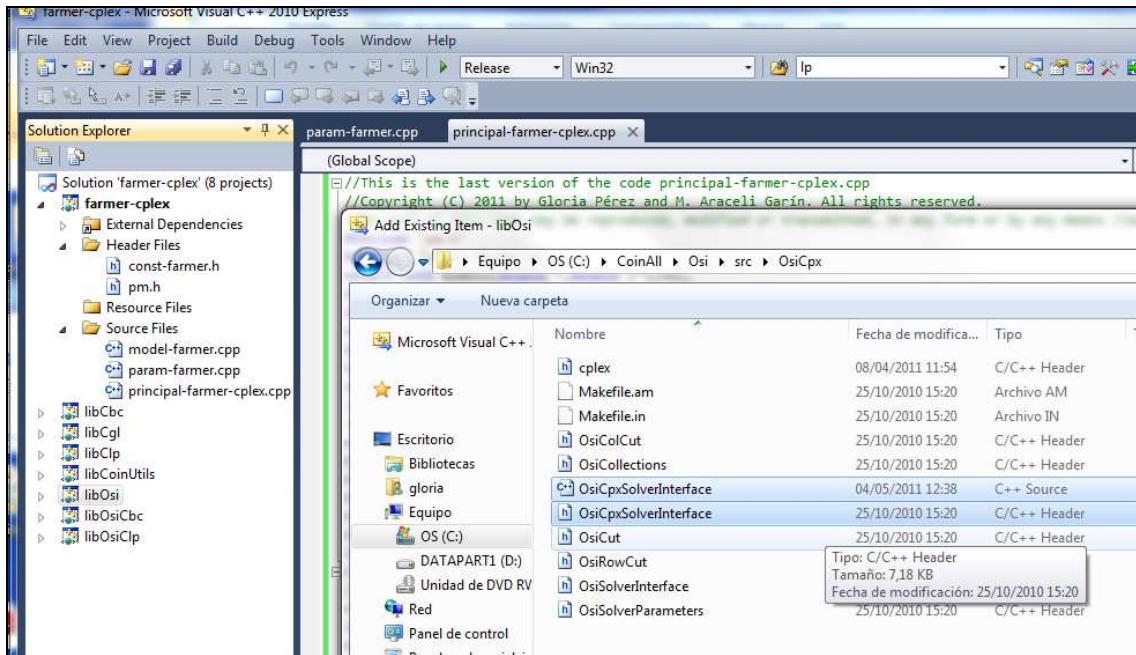
After doing it, you must click in the File button, and add as Existing seven COIN projects. You must look for the path to the directory where is each of them. The seven paths and projects are:

- C:\CoinAll\Cbc\MSVisualStudio\v9\libCbc\libCbc.vcproj
- C:\CoinAll\Cgl\MSVisualStudio\v9\libCgl\libCgl.vcproj
- C:\CoinAll\Clp\MSVisualStudio\v9\libClp\libClp.vcproj
- C:\CoinAll\CoinUtils\MSVisualStudio\v9\libCoinUtils\libCoinUtils.vcproj
- C:\CoinAll\Osi\MSVisualStudio\v9\libOsi\libOsi.vcproj
- C:\CoinAll\Osi\MSVisualStudio\v9\libOsiCbc\libOsiCbc.vcproj
- C:\CoinAll\Osi\MSVisualStudio\v9\libOsiClp\libOsiClp.vcproj

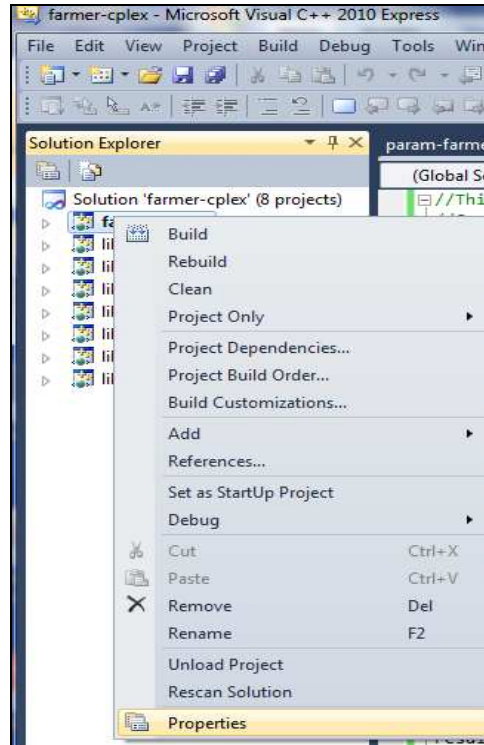
Then click on the file with extension .vcproj to open and add it. After doing it, the screen look likes this. Notice in the left hand menu, where are the added projects.



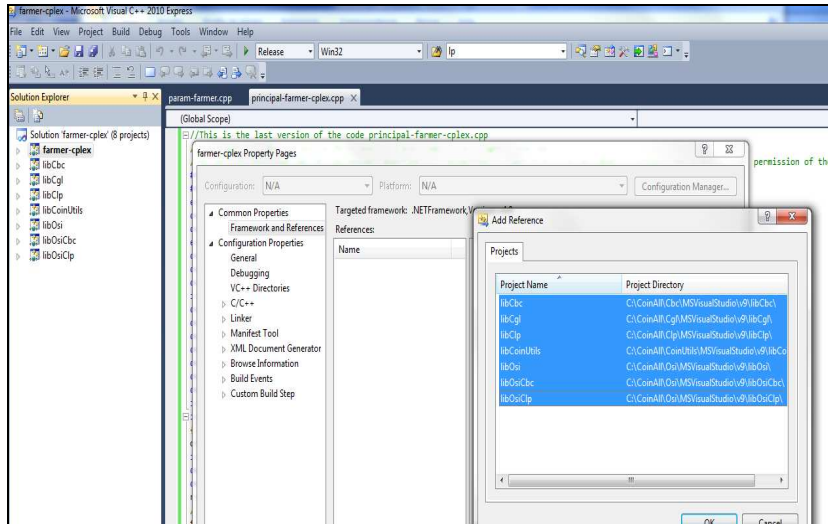
In the same way you have to add in libOsi from the directory C:\CoinAll\Osi\src\OsiCpx and as Existing items, the files OsiCpxSolverInterface.hpp and OsiCpxSolverInterface.cpp.



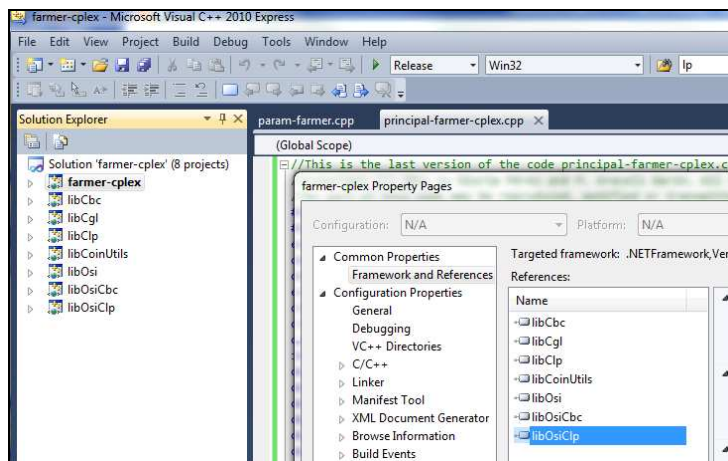
Finally, you must modify some of the properties in you project in order to compile your code with the COIN-OR source code and link with the IBM ILOG CPLEX libraries. To do it, click on the right button beside the boldface name of the project, **farmer-cplex**, and select the last item, Properties.



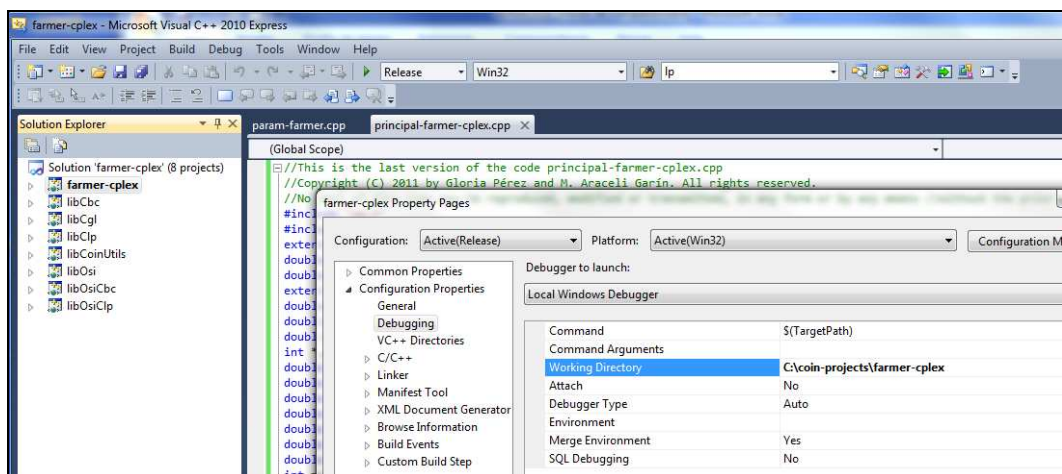
In the left menu, in Common Properties you must add seven new references, clicking on the Add New Reference button, and select the seven projects: libCbc, libCgl, libClp, libCoinUtils, libOsi, libOsiCbc and libOsiClp.



The screen becomes as follows.



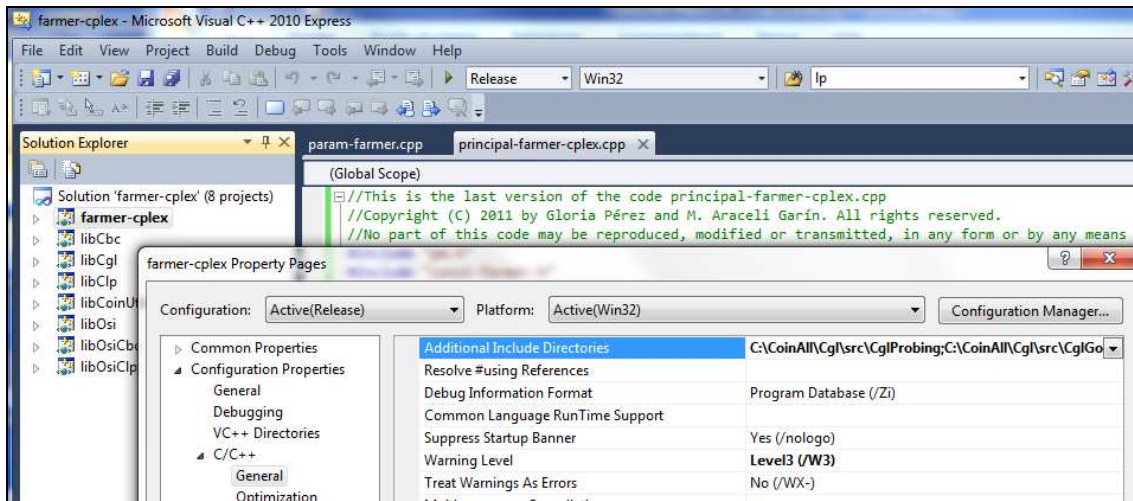
In the Debugging options, you must establish the working directory, C:\coin-projects\farmer-cplex, and click on Aplicar button.



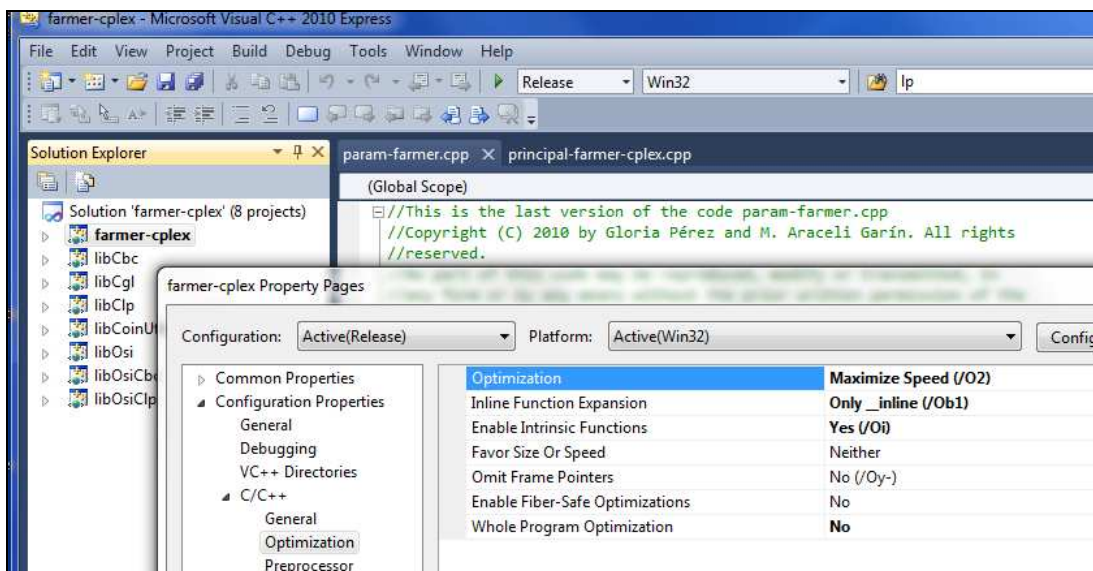
Then, select the C/C++ and General button, and write as Additional include directions the following fourteen directories, all of them separated by semi-colons:

```
C:\CoinAll\Cgl\src\CglProbing; C:\CoinAll\Cgl\src\CglGomory;
C:\CoinAll\Cgl\src\CglClique; C:\CoinAll\Cgl\src\CglKnapsackCover;
C:\CoinAll\Cgl\src; C:\CoinAll\Clp\src;
C:\CoinAll\Cbc\src; C:\CoinAll\Osi\src;
C:\CoinAll\BuildTools\headers; C:\CoinAll\Osi\src\OsiCbc;
C:\CoinAll\Osi\src\OsiClp; C:\CoinAll\CoinUtils\src;
C:\CoinAll\Osi\src\OsiCpx;
C:\ILOG\CPLEX_Studio_AcademicResearch122\cplex\include\ilcplex;
```

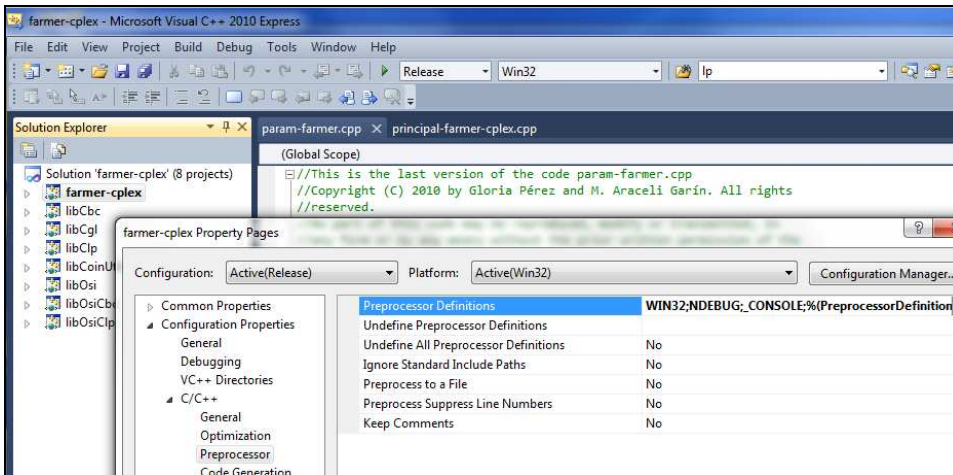
These are the directories with the header files included in the header file *pm.h*.



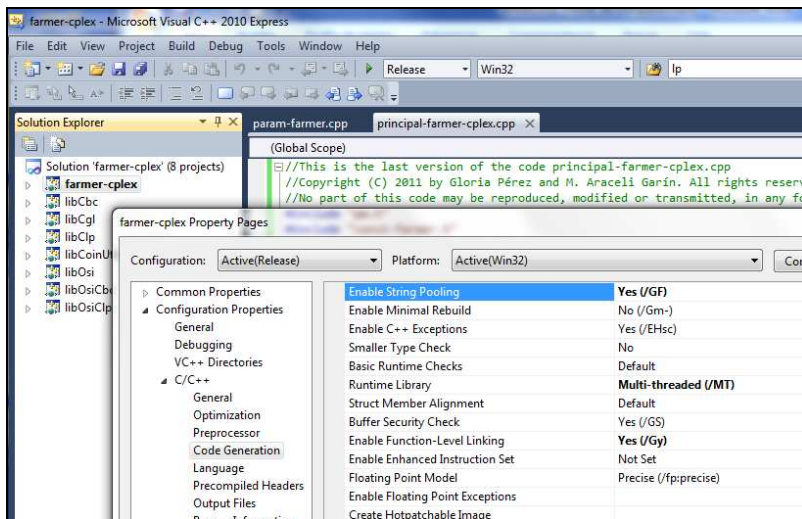
In the Optimization options, set as follows and click on the Aplicar button.



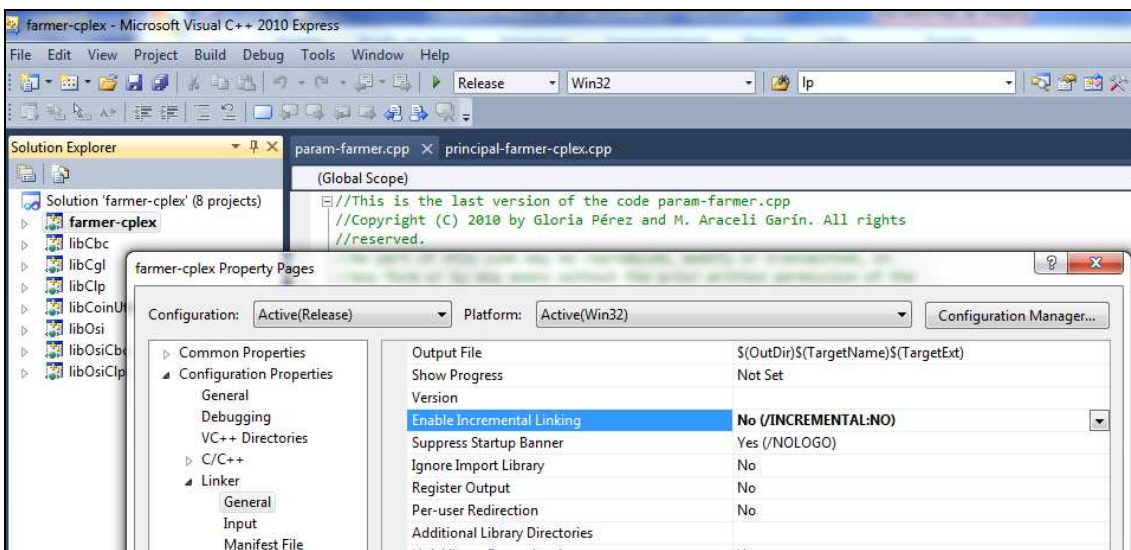
In Preprocessor, select the following definitions: WIN32; NDEBUG; _CONSOLE; and _CRT_SECURE_NODEPRECATE; and click on the Aplicar button.



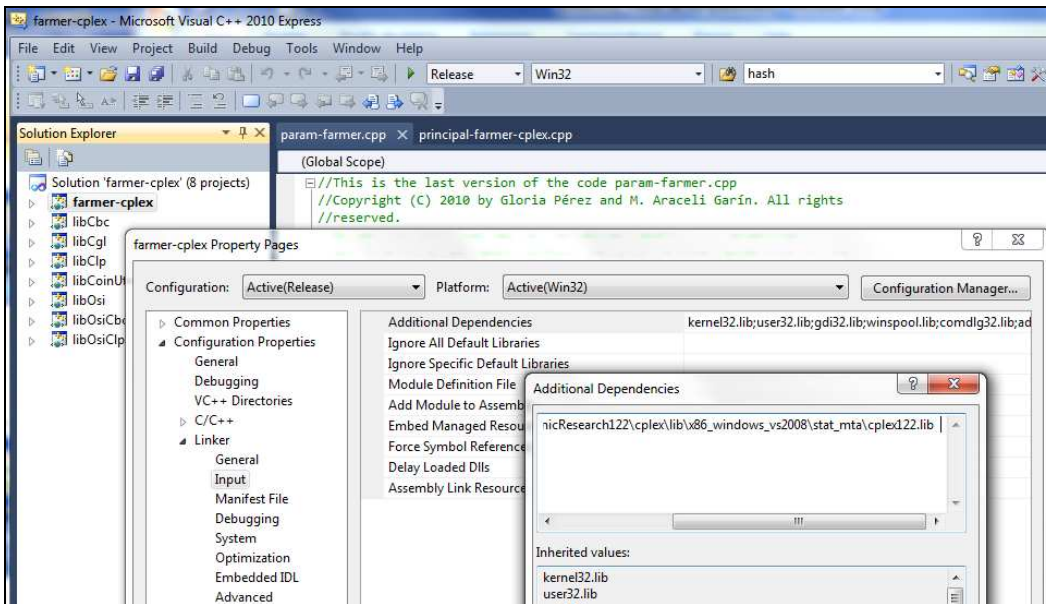
In Code Generation, set as follows and click on the Aplicar button.



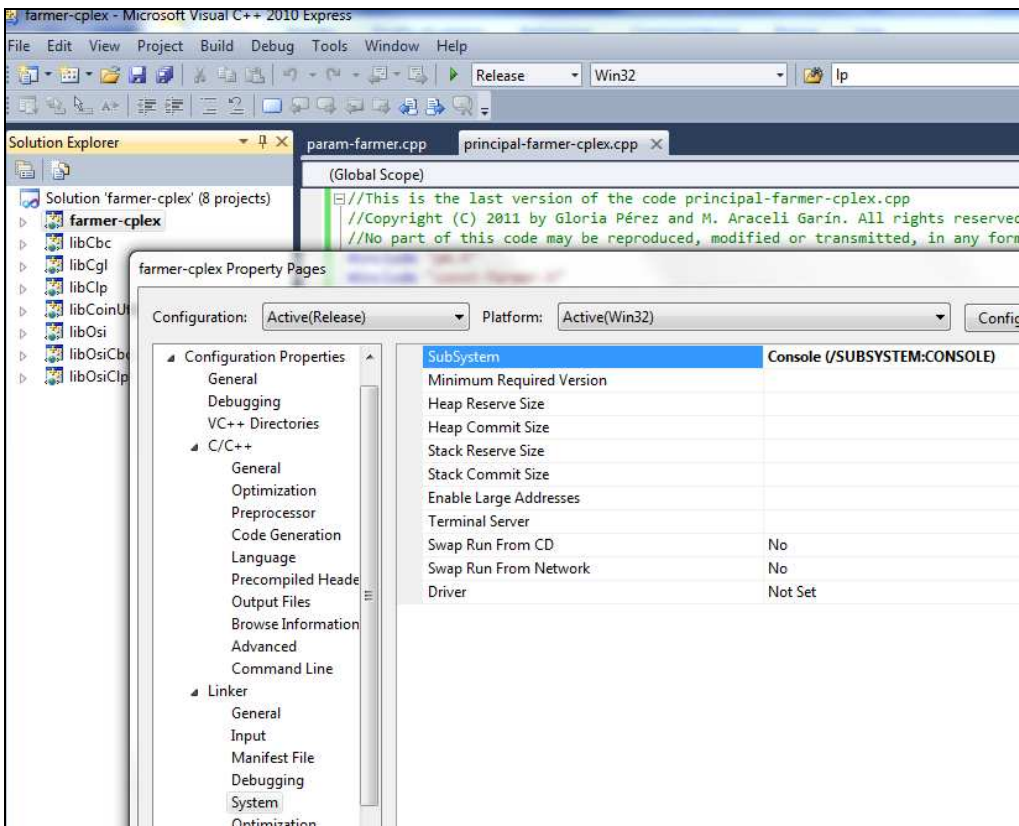
In Linker, click on General and set the following options. Then, click on Aplicar button.



In Input button, select Additional Dependencies and add the location
 C:\ILOG\CPLEX_Studio_AcademicResearch122\cplex\lib\x86_windows_vs2008\stat_mta\cplex122.lib
 for Windows 32bits.



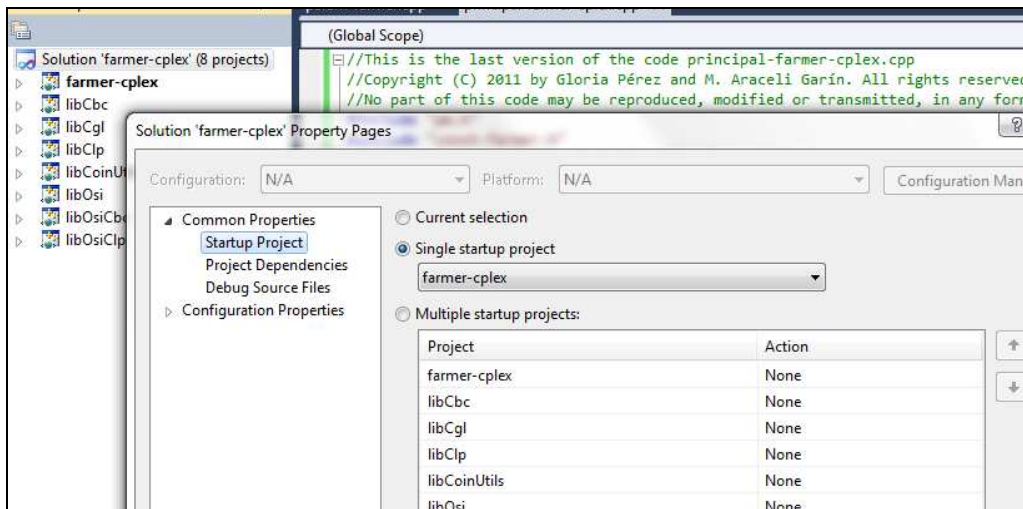
In System, set the following options and click on the Aplicar button:



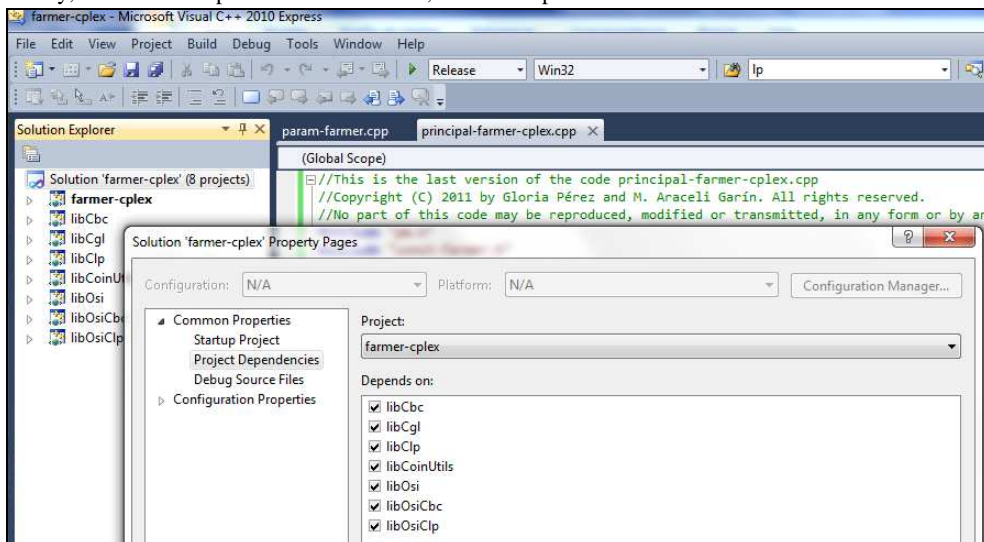
Finally, click on the Aceptar button to go to the start project screen.

Now you must modify some properties of your solution (executable). To do it, click on the right button over the icon beside Solution `farmer-cplex` (8 projects), and select the last option, Properties.

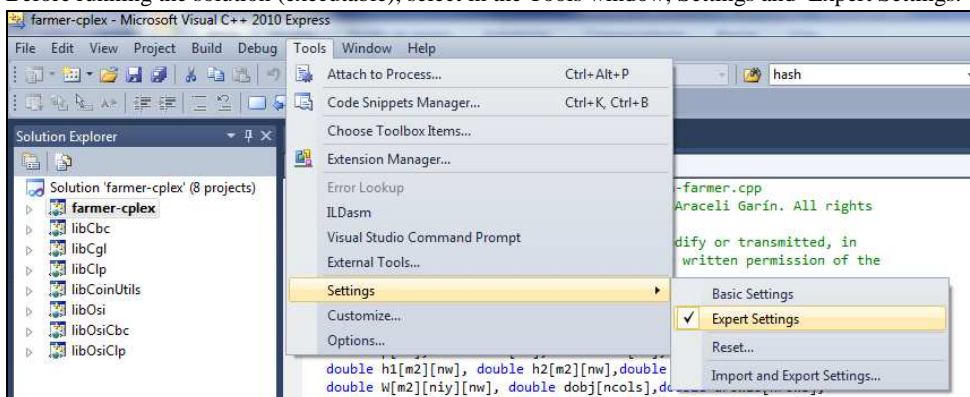
In the left screen, select Startup Project in the Common Properties, and in the right screen click on Single Startup project you have to have selected *farmer-cplex* project, like in the screen.



To set the Project Dependencies, select in the right hand screen **farmer-cplex** as Project, and click on all the projects given below. Finally, click on the Aplicar button and then, on the Aceptar button.



Before running the solution (executable), select in the Tools window, Settings and Expert Settings.



Now you can compile and link your code, i.e., you can build the solution, *farmer-cplex.sln*. To do it, in the start screen, click on the Build button and then, on Build Solution. If all went fine, you obtain 0 failed, and the Solution (executable) has been built. This information appears in the bottom of the screen. The screen look likes this:

```

//STEP 0. MODEL GENERATION
tiempo0=CoinCpuTime();
results<<"CPU time for loading data model "<<CoinCpuTime()<<"\n";
models(c1,c2,p,b1,b2,A,h1,h2,T,W);
//STEP 1. INTRODUCTION OF THE COEFFICIENTS IN THE ARRAYS OF COIN
nocero=0;
param(c1,c2,p,b1,b2,A,h1,h2,T,W,dobj,drowlo,
drowup,dcollo,dcolup,nrowindx,ncolindx,dels,nocero);
//STEP 2. DEFINE THE MODEL IN COIN-OR
//without using interface (OSIClpSolverinterface)
// ClpSimplex *soll; soll=new ClpSimplex;
//or alternatively, by using interface
100%

```

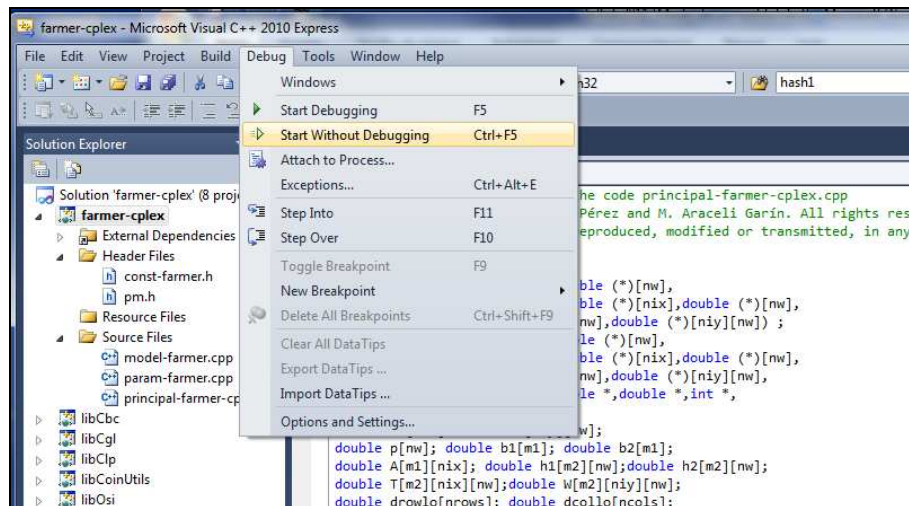
Output

```

Show output from: Build
2>Project not selected to build for this solution configuration
3>----- Skipped Build: Project: libCoinUtils, Configuration: Release Win32 -----
3>Project not selected to build for this solution configuration
4>----- Skipped Build: Project: libClp, Configuration: Release Win32 -----
4>Project not selected to build for this solution configuration
5>----- Skipped Build: Project: libCgl, Configuration: Release Win32 -----
5>Project not selected to build for this solution configuration
6>----- Skipped Build: Project: libCbc, Configuration: Release Win32 -----
6>Project not selected to build for this solution configuration
7>----- Skipped Build: Project: libOsiClp, Configuration: Release Win32 -----
7>Project not selected to build for this solution configuration
===== Build: 0 succeeded, 0 failed, 1 up-to-date, 7 skipped =====

```

To run the solution and get the output file, go to the start screen and click on Debug, and then on Start without Debugging, in a screen like this:



After that, you get a black screen and the output file *resul-farmer.dat* has been generated in your working directory.

```

C:\Windows\system32\cmd.exe
Tried aggregator 1 time.
LP Presolve eliminated 3 rows and 6 columns.
Reduced LP has 10 rows, 21 columns, and 30 nonzeros.
Presolve time = 0.00 sec.

Iteration log . . .
Iteration: 1 Scaled infeas = 239.999998
Switched to dexev.
Iteration: 3 Objective = 20500.000000
Found feasible solution after 0.00 sec. Objective = 98000.0000
Probing time = 0.00 sec.
MIP emphasis: balance optimality and feasibility.
MIP search method: dynamic search.
Parallel mode: deterministic, using up to 8 threads.
Root relaxation solution time = 0.01 sec.

Nodes
Node Left Objective IInf Best Integer Cuts/ Best Node ItCnt Gap
* 0+ 0 98000.0000 0 ---
* 0+ 0 -14150.0000 0 ---
* 0+ 0 -19100.0000 0 ---
* 0+ 0 -24050.0000 0 ---
* 0+ 0 -29000.0000 0 ---
* 0+ 0 -33950.0000 0 ---
* 0+ 0 -38900.0000 0 ---
* 0+ 0 -43850.0000 0 ---
* 0+ 0 -48800.0000 0 ---
* 0+ 0 -53750.0000 0 ---
* 0 0 integral 0 -108390.0000 -108390.0000 0 0.00%
Elapsed real time = 0.01 sec. (tree size = 0.00 MB, solutions = 11)

Root node processing (before b&c):
Real time = 0.01
Parallel b&c, 8 threads:
Real time = 0.00
Sync time (average) = 0.00
Wait time (average) = 0.00

Total (root+branch&cut) = 0.01 sec.
Default column names x1, x2 ... being created.
Default row names c1, c2 ... being created.
Presione una tecla para continuar . . .

```

5. The download process and basic installation of CPLEX within COIN-OR under Linux-like systems

Again, the first step is to download the source code of COIN-OR. In similar way as is described in Section 3, for doing it, you must click on [Download/Use](#) in the left hand side of the home page <http://www.coin-or.org>. Then in the second Section titled Source Code, you must click on [here](#). You can observe an index for the source of a number of COIN projects. You must click on [CoinAll/](#) to obtain the list of last versions. In this case you will click and select [CoinAll-1.4.0.tgz](#). Alternatively, you can go directly to the home page for this version, <http://www.coin-or.org/download/source/CoinAll/CoinAll-1.4.0.tgz>. This project will only build in Linux-like environments using the GNU autotools. The compiler `gcc v4.1` at least will be needed. After downloading the tarball you must extract the code, for example, typing in the prompt

```
$ tar xzvf CoinAll-1.4.0.tgz
```

This sentence creates a subdirectory (by default named `CoinAll-1.4.0`), that you can rename as `CoinAll`, where is the source code. You can also by clicking on the right button over the `.tgz`, extract the code into. Then, go to the directory that you just downloaded or extracted (in our case, `CoinAll`) and type the following script,

```
$ ./configure COIN_SKIP_PROJECTS=`Smi Alps Bcp Bcps Blis Thirdparty SYMPHONY`
```

With this script, the projects between ``` and ``` are not installed. They are not needed for solving linear and/or integer optimization problems. If everything went fine, you will see at the end of the output “Main configuration of CoinAll successful”

In the directory where you ran the configure script, you must install the code. To do it, you type

```
$ make install
```

After this, you will find the executables, libraries and header files in the “bin”, “lib” and “include” subdirectories, respectively. Now you can compile and link your source code with the COIN-OR solvers.

As in the case of Windows systems, if we want to use the solvers of IBM ILOG CPLEX, in a second step we must download the CPLEX libraries with academic license. The full process has been detailed in Section 3 for Windows systems and now, we will present the main steps in Linux environments. At the beginning, you must go to the home page <http://www-01.ibm.com/software/websphere/products/optimization/academic-initiative/index.html>

In the right hand side, middle page and under the epigraph Membership, you must click on [Join now](#), for registration. After doing it, you must complete process to become a member of the Academic Initiative. In the first step click on [Register](#), and complete the required information. You must give your institutional (UPV/EHU, if case) e-mail address, and a password, and click on [Sign in](#) button. After obtaining your membership, you can access to the IBM Academic Initiative resources. You can back to the IBM Academic Initiative home page, and click on the second option [Get full-version software](#) of the epigraph Member offerings under the Membership epigraph. In the following screen, select the [Download from the Software Catalog](#) option in the middle of the page. You will see the page entitled IBM Academic Initiative Program, where you must click on the [Submit](#) button if you are agree that IBM may process your data. After submitting, you go to the following page and in the left hand side of the screen you must click on the first item [Search for software](#). You must type CPLEX in the box of Find by search text. Before clicking on [Search](#), to filter your search, type in the box of Find by part number, CZJS3ML and then, click on the search button. The selected version must be IBM ILOG CPLEX Optimization Studio Research Edition v12.2 for Linux x86 and x86-64 Multilingual. Click also on I agree, and at the end of the screen, click on the [Download now](#) button. By default, the software has been downloaded in the directory named DownloadDirector. In this location, there are two files, [cplex_studio122.acad.linux-x86.bin](#) and [dlmgr.pro](#). You must type

```
$ chmod 777 cplex_studio122.acad.linux-x86.bin
```

to get the executable file, and then, type

```
$ ./cplex_studio122.acad.linux-x86.bin
```

to start the installation in the directory ILOG. When the installation is completed, the message in the screen is:
IBM ILOG CPLEX Optimization Studio Academic Research Edition 12.2 has been successfully installed to:
/home/ILOG/CPLEX_Studio_AcademicResearch122

IBM ILOG CPLEX is a product that needs a license key. To get it, go back to the home page from where you have downloaded the code and click on [ILOG support](#), in the paragraph [Note about license keys](#).

You must follow step by step the process for obtaining the license key. Click on the [ILOG Optimization Key Request](#) button at Step 2. As membership, with your ID and password, you will obtain the following screen. Click on [Download now](#) button, and you will get a plain text file, the key file access.ilm. Follow the guide [ILOGQuickStart.pdf](#) for its installation. To obtain this file, please click on [Quick Start guide](#) button. Save a copy of the license key file **access.ilm** in the location /home/ILOG/ilm. You must create a link with the directory /usr/ilog. To do it, type

```
$ ln -s /home/ILOG /usr/ilog
```

Now you must create the library OsiCpxSolverInterface.lo. To do it, go to the locations /CoinAll/Cbc and /CoinAll/Osi and type in both of them,

```
$ ./configure  
$ make  
$ make install
```

Look at the directory /ILOG/CPLEX_Studio_AcademicResearch122/cplex/include/ilcplex the file **cplex.h**. Copy this file into the directory /CoinAll/Osi/src/OsiCpx.

Go to the directory /CoinAll/Osi/src, and locate the following six header files: **OsiSolverInterface.hpp**, **OsiCollections.hpp**, **OsiSolverParameters.hpp**, **OsiCut.hpp**, **OsiRowCut.hpp**, and **OsiColCut.hpp**. Copy all of them to the subdirectory /CoinAll/Osi/src/OsiCpx. Now, go to the location /CoinAll/Osi/src/OsiCpx and type,

```
$ make
```

Now in this directory the library OsiCpxSolverInterface.lo has been created. Copy it to the directory /CoinAll/Cbc/lib/

Finally, you must add two environment user variables in your computer. To do it, you can edit the hidden file .bashrc, and add at the end of this file both definitions as follows

```
ILOG_LICENSE_FILE= "/home/ILOG/ILM/access.ilm"
```

PATH= \$PATH; /home/ILOG/CPLEX_Studio_AcademicResearch122/cplex/bin/x86_sles10_4.1

Now, you must save the file and restart your computer.

6. Linking your code with CPLEX within COIN-OR and running the executable under Linux

Assume that you have downloaded and installed the COIN-OR sources in the directory CoinAll, and downloaded and installed the CPLEX code in the directory /ILOG/CPLEX_Studio_AcademicResearch122/, and made all the steps in the join installation procedure described in Section 5.

For most COIN-OR packages the main directory contains an example subdirectory. Assuming that this is the case for the package Cbc, the directory CoinAll/Cbc/Examples contains a Makefile that has been adapted to your system, see Appendix B for the Makefile modified.

Copy this Makefile in your working directory where you have edited your own code, i.e., the files *const-farmer.h*, *model-farmer.cpp*, *param-farmer.cpp*, *principal-farmer-cplex.cpp* and *pm.h*. In order to modify this Makefile to compile your own code, you only have to change some things. Edit the file Makefile, where you put the name of the executable, which in the example is "driver" now you must write a name for our executable, for example "farmer-cplex". You must change a set of sentences that you can look in the Makefile given in Appendix B.

From the prompt of your working directory, type
\$ make -k farmer-cplex

to compile and link the code. If all went fine, an executable named *farmer-cplex* exists in your directory. To run it, you must type
\$./farmer-cplex

Again, if all went fine you will see the following output, IBM ILOG License Manager: "IBM ILOG Optimization Suite for Academic Initiative" is accessing CPLEX 12 with option(s): "e m b q".

```
Tried aggregator 1 time.
LP Presolve eliminated 3 rows and 6 columns.
Reduced LP has 10 rows, 21 columns, and 30 nonzeros.
Presolve time = 0.00 sec.
Iteration log . . .
Iteration: 1 Scaled infeas = 239.999998
Switched to dexv.
Iteration: 3 Objective = 20500.000000
Found feasible solution after 0.02 sec. Objective = 98000.0000
Probing time = 0.00 sec.
MIP emphasis: balance optimality and feasibility.
MIP search method: dynamic search.
Parallel mode: deterministic, using up to 4 threads.
Root relaxation solution time = 0.04 sec.
Nodes Cuts/

Node Left Objective IInf Best Integer Best Node ItCnt Gap
* 0+ 0 98000.0000 0 ---
* 0+ 0 -14150.0000 0 ---
* 0+ 0 -19100.0000 0 ---
* 0+ 0 -24050.0000 0 ---
* 0+ 0 -29000.0000 0 ---
* 0+ 0 -33950.0000 0 ---
* 0+ 0 -38900.0000 0 ---
* 0+ 0 -43850.0000 0 ---
* 0+ 0 -48800.0000 0 ---
* 0+ 0 -53750.0000 0 ---
* 0 0 integral 0 -108390.0000 -108390.0000 0 0.00%

Elapsed real time = 0.09 sec. (tree size = 0.00 MB, solutions = 11)
Root node processing (before b&c):
Real time = 0.08
```



```
Parallel b&c, 4 threads:
Real time      = 0.00
Sync time (average) = 0.00
Wait time (average) = 0.00
-----
```

```
Total (root+branch&cut) = 0.08 sec.
Default column names x1, x2 ... being created.
Default row names c1, c2 ... being created.
```

Notice that the two last rows in the file `principal-farmer-cplex.cpp` must be commented for the Linux compilation. After the execution, in your directory there exists a new file, `resul-farmer.dat`, with the output of the execution, see Appendix C.

You can also use an editor like emacs, that allows you to compile C++ code moreover than latex code. In this case, to run the executable, you can do it from the prompt of the working directory where the executable is.

Appendix A

A.1 File `const-farmer.h`

```
//This is the last version of the code const-farmer.h
//Copyright (C) 2011 by Gloria Pérez and M. Araceli Garín. All rights reserved. No part of this code
//may be reproduced, modified or transmitted, in any form or by any means without the prior written permission of
//the authors. Integer constants which define the dimensions of the stochastic version of the farmer' problem.
//Model (1.5), pp.11, Birge and Louveaux (1997)
//Number of scenarios
#define nw 3
//Number of first stage continuous variables
#define nix 3
//Number of second stage continuous variables
#define niy 6
//Number of first stage constraints
#define m1 1
//Number of second stage constraints
#define m2 4
//Number of variables and constraints of the whole problem
#define ncols nix+niy*nw
#define nrows m1+m2*nw
#define nelement m1*nix+m2*(nix+niy)*nw
```

A.2 File `pm.h`

```
//This is the last version of the code pm.h
//Copyright (C) 2011 by Gloria Pérez and M. Araceli Garín. All rights reserved.
//No part of this code may be reproduced, modified or transmitted, in any form or by any means
//without the prior written permission of the authors.
#include <string.h>
#include <stdlib.h>
#include <math.h>
#include <assert.h>
#include <stdio.h>
#include <iostream>
#include <fstream>
using namespace std;
#include <cstdlib>
#include "ClpSimplex.hpp"
#include "CoinHelperFunctions.hpp"
#include "CoinTime.hpp"
#include "CoinBuild.hpp"
#include "CoinModel.hpp"
#include "OsiClpSolverInterface.hpp"
#include "CbcModel.hpp"
#include "CoinPackedMatrix.hpp"
#include "CglKnapsackCover.hpp"
#include "OsiCuts.hpp"
```

```

#include "CglClique.hpp"
#include "CglGomory.hpp"
#include "CglProbing.hpp"
//cplex
#include "cplex.h"
#include "OsiCpxSolverInterface.hpp"

```

A.3 File *model-farmer.cpp*

```

//This is the last version of the code model-farmer.cpp
//Copyright (C) 2011 by Gloria Pérez and M. Araceli Garín. All rights reserved.
//No part of this code may be reproduced, modified or transmitted, in any form or by any means
//without the prior written permission of the authors.
#include "pm.h"
#include "const-farmer.h"

void models(double c1[nix],double c2[niy][nw],
            double p[nw], double b1[m1], double b2[m1],
            double A[m1][nix], double h1[m2][nw],
            double h2[m2][nw],double T[m2][nix][nw],
            double W[m2][niy][nw])

//Model coefficients of farmer' problem. Birge and Louveaux(1997),pp.4-15
{
    int i,iomega;
//weights p
    for (iomega=0; iomega<nw; iomega++) p[iomega]=1.0/(1.0*nw);
//obj coefficients c1,c2
    c1[0]=150.;
    c1[1]=230.;
    c1[2]=260.;
//obj coeficientes WITHOUT weights
    for (iomega=0;iomega<nw;iomega++)
    {
        c2[0][iomega]=238.;
        c2[1][iomega]=210.;
        c2[2][iomega]=-170.;
        c2[3][iomega]=-150.;
        c2[4][iomega]=-36.;
        c2[5][iomega]=-10.;
    }
//left and right hand sides: b1,b2, h1 and h2
    b1[0]=-1e31;
    b2[0]=500;
    for (iomega=0;iomega<nw;iomega++)
    {
        h1[0][iomega]=-1e31; h2[0][iomega]=-200.;
        h1[1][iomega]=-1e31; h2[1][iomega]=-240.;
        h1[2][iomega]=-12000.; h2[2][iomega]=0.0;
        h1[3][iomega]=0.; h2[3][iomega]=6000.0;
    }

//matrix A, x variables
    for (i=0;i<nix;i++) A[0][i]=1.0;
// matrices T and W
    //scenario 1
    iomega=0;
// T[j-1][i-1][iomega]: j=1,m2;i=1,nix;iomega=1,nw
T[0][0][iomega]=-3.;T[0][1][iomega]=0.; T[0][2][iomega]=0.; //equation 1
T[1][0][iomega]=0.; T[1][1][iomega]=-3.6; T[1][2][iomega]=0.;//equation 2
T[2][0][iomega]=0.; T[2][1][iomega]=0.; T[2][2][iomega]=-24.;//equation 3
T[3][0][iomega]=0.; T[3][1][iomega]=0.; T[3][2][iomega]=0.; //equation 4

```

```

//W[j-1][i-1][iomega]:j=1,m2;i=1,niy;iomega=1,nw
//equation 1
W[0][0][iomega]=-1.://y_1
W[0][1][iomega]=0.://y_2
W[0][2][iomega]=1.://w_1
W[0][3][iomega]=0.://w_2
W[0][4][iomega]=0.://w_3
W[0][5][iomega]=0.://w_4
//equation 2
W[1][0][iomega]=0.;
W[1][1][iomega]=-1.;
W[1][2][iomega]=0.;
W[1][3][iomega]=1.;
W[1][4][iomega]=0.;
W[1][5][iomega]=0.;
//equation 3
W[2][0][iomega]=0.;
W[2][1][iomega]=0.;
W[2][2][iomega]=0.;
W[2][3][iomega]=0.;
W[2][4][iomega]=1.;
W[2][5][iomega]=1.;
//equation 4
W[3][0][iomega]=0.;
W[3][1][iomega]=0.;
W[3][2][iomega]=0.;
W[3][3][iomega]=0.;
W[3][4][iomega]=1.;
W[3][5][iomega]=0;
//*****
//scenario 2
iomega=1;
//*****
// T[j-1][i-1][iomega]: j=1,m2;i=1,nix;iomega=1,nw

T[0][0][iomega]=-2.5; T[0][1][iomega]=0.; T[0][2][iomega]=0.; //equation 1
T[1][0][iomega]=0.; T[1][1][iomega]=-3.; T[1][2][iomega]=0.; //equation 2
T[2][0][iomega]=0.; T[2][1][iomega]=0.; T[2][2][iomega]=-20.; //equation 3
T[3][0][iomega]=0.; T[3][1][iomega]=0.; T[3][2][iomega]=0.; //equation 4

//W[j-1][i-1][iomega]:j=1,m2;i=1,niy;iomega=1,nw
//equation 1
W[0][0][iomega]=-1.://y_1
W[0][1][iomega]=0.://y_2
W[0][2][iomega]=1.://w_1
W[0][3][iomega]=0.://w_2
W[0][4][iomega]=0.://w_3
W[0][5][iomega]=0.://w_4
//equation 2
W[1][0][iomega]=0.;
W[1][1][iomega]=-1.;
W[1][2][iomega]=0.;
W[1][3][iomega]=1.;
W[1][4][iomega]=0.;
W[1][5][iomega]=0.;
//equation 3
W[2][0][iomega]=0.;
W[2][1][iomega]=0.;
W[2][2][iomega]=0.;
W[2][3][iomega]=0.;
W[2][4][iomega]=1.;
W[2][5][iomega]=1.;
//equation 4

```

```

W[3][0][iomega]=0.;
W[3][1][iomega]=0.;
W[3][2][iomega]=0.;
W[3][3][iomega]=0.;
W[3][4][iomega]=1.;
W[3][5][iomega]=0;
//scenario 3
iomega=2;
//T[j-1][i-1][iomega]: j=1,m2;i=1,nix;iomega=1,nw
T[0][0][iomega]=-2.; T[0][1][iomega]=0.; T[0][2][iomega]=0.; //equation 1
T[1][0][iomega]=0.; T[1][1][iomega]=-2.4; T[1][2][iomega]=0.; //equation 2
T[2][0][iomega]=0.; T[2][1][iomega]=0.; T[2][2][iomega]=-16.; //equation 3
T[3][0][iomega]=0.; T[3][1][iomega]=0.; T[3][2][iomega]=0.; //equation 4

//W[j-1][i-1][iomega]:j=1,m2;i=1,niy;iomega=1,nw
//equation 1
W[0][0][iomega]=-1.;//y_1
W[0][1][iomega]=0.;//y_2
W[0][2][iomega]=1.;//w_1
W[0][3][iomega]=0.;//w_2
W[0][4][iomega]=0.;//w_3
W[0][5][iomega]=0.;//w_4
//equation 2
W[1][0][iomega]=0.;
W[1][1][iomega]=-1.;
W[1][2][iomega]=0.;
W[1][3][iomega]=1.;
W[1][4][iomega]=0.;
W[1][5][iomega]=0.;
//equation 3
W[2][0][iomega]=0.;
W[2][1][iomega]=0.;
W[2][2][iomega]=0.;
W[2][3][iomega]=0.;
W[2][4][iomega]=1.;
W[2][5][iomega]=1.;
//equation 4
W[3][0][iomega]=0.;
W[3][1][iomega]=0.;
W[3][2][iomega]=0.;
W[3][3][iomega]=0.;
W[3][4][iomega]=1.;
W[3][5][iomega]=0.;
}

```

A.4 File *param-farmer.cpp*

```

//This is the last version of the code param-farmer.cpp
//Copyright (C) 2011 by Gloria Pérez and M. Araceli Garín. All rights reserved.
//No part of this code may be reproduced, modified or transmitted, in any form or by any means
//without the prior written permission of the authors.
#include "pm.h"
#include "const-farmer.h"

void param(double c1[nix],double c2[niy][nw],
double p[nw], double b1[m1], double b2[m1], double A[m1][nix],
double h1[m2][nw], double h2[m2][nw],double T[m2][nix][nw],
double W[m2][niy][nw], double dobj[ncols],double drowlo[nrows],
double drowup[nrows],double dcollo[ncols],double dcolup[ncols],
int nrowindx[nelement],int mcolindx[nelement],
double dels[nelement],int &nocero)
{
int Newnelement=0; int Newnrows=0; int Newncols=0;
int i,iomega,j;

```

```

//Matrix A: first stage matrix coefficients
if(m1!=0)
{
for (j=0;j<m1;j++)
{
for (i=0;i<nix;i++)
{
dels[Newnelement]=A[j][i];
mcolindx[Newnelement]=i;
dobj[mcolindx[Newnelement]]=c1[i];
dcollo[mcolindx[Newnelement]]=0.;
dcolup[mcolindx[Newnelement]]=1e31;
nrowindx[Newnelement]=j;
Newnelement++;
}
}
drowlo[Newnrows]=b1[j];
drowup[Newnrows]=b2[j];
Newnrows++;
}
}
Newncols=nix;

//Matrices T and W: second stage matrix coefficients
// x-variables, Matrix T
if(m2!=0){
for (iomega=0;iomega<nw;iomega++)
{
for (j=0;j<m2;j++)
{
for (i=0;i<nix;i++)
{
dels[Newnelement]=T[j][i][iomega];
mcolindx[Newnelement]=i;
nrowindx[Newnelement]=Newnrows;
dobj[mcolindx[Newnelement]]=c1[i];
dcollo[mcolindx[Newnelement]]=0.;
dcolup[mcolindx[Newnelement]]=1e31;
Newnelement++;
}
}
}
// y-variables, Matrix W
for (i=0;i<niy;i++)
{
dels[Newnelement]=W[j][i][iomega];
mcolindx[Newnelement]=Newncols+i;
nrowindx[Newnelement]=Newnrows;
dobj[mcolindx[Newnelement]]=c2[i][iomega]*p[iomega];
dcollo[mcolindx[Newnelement]]=0.;
dcolup[mcolindx[Newnelement]]=1e31;
Newnelement++;
}
drowlo[Newnrows]=h1[j][iomega];
drowup[Newnrows]=h2[j][iomega];
Newnrows++; //Total constraints
}
Newncols=Newncols+niy; //Total variables
}
}
nocero=Newnelement; //Total nonzero elements
}

```

A.5 File *principal-farmer-cplex.cpp*

```
//This is the last version of the code principal-farmer-cplex.cpp
//Copyright (C) 2011 by Gloria Pérez and M. Araceli Garín. All rights reserved.
//No part of this code may be reproduced, modified or transmitted, in any form or by any means
//without the prior written permission of the authors.
#include "pm.h"
#include "const-farmer.h"
extern void models(double *,double (*)(nw),
double *,double *,double *, double (*)(nix),double (*)(nw),
double (*)(nw),double (*)(nix)[nw],double (*)(niy)[nw]) ;
extern void param(double *,double (*)(nw),
double *,double *,double *, double (*)(nix),double (*)(nw),
double (*)(nw),double (*)(nix)[nw],double (*)(niy)[nw],
double *,double *,double *,double *,double *,int *,
int *,double *,int &);
double c1[nix];double c2[niy][nw];
double p[nw]; double b1[m1]; double b2[m1];
double A[m1][nix]; double h1[m2][nw];double h2[m2][nw];
double T[m2][nix][nw];double W[m2][niy][nw];
double drowlo[nrows]; double dcollo[ncols];
double drowup[nrows]; double dcolup[ncols];
double dobj[ncols],dels[nelement];
int mcolindx[nelement]; int nrowindx[nelement];
int main()
{
ofstream results("resul-farmer.dat"); //output file
int i,iomega,j,nocero;
double tiempo1, tiempo0, tiempo01;
double p[nw];
results<<"Farmer' Problem: OUTPUT \n";
//STEP 0. MODEL GENERATION
tiempo0=CoinCpuTime();
results<<"CPU time for loading data model "<<CoinCpuTime()<<"\n";
models(c1,c2,p,b1,b2,A,h1,h2,T,W);
//STEP 1. INTRODUCTION OF THE COEFFICIENTS IN THE ARRAYS OF COIN
nocero=0;
param(c1,c2,p,b1,b2,A,h1,h2,T,W,dobj,drowlo,
drowup,dcollo,dcolup,nrowindx,mcolindx,dels,nocero);
//STEP 2. DEFINE THE MODEL IN CPLEX within COIN-OR
OsiCpxSolverInterface sol1;
//Load the matrix coefficients by indices or alternatively, you can
//read the data from a .mps file
// In this case you do not need models and param functions
// sol1.readMps("model-farmer.mps");
CoinPackedMatrix AA(true,nrowindx,mcolindx,dels,nocero);
sol1.loadProblem(AA,dcollo,dcolup,dobj,drowlo,drowup);
results<<"CPU input time COIN "<<CoinCpuTime()<<"\n";
tiempo01=CoinCpuTime();
results<<"Number of variables:"<<sol1.getNumCols()<<"\n";
results<<"Number of constraints:"<<sol1.getNumRows()<<"\n";
results<<"Number of nonzero elements:"<<nocero<<"\n";
double dens=(nelement*100.0)/((ncols*1.0)*(nrows*1.0));
results<<"Matrix density:"<<dens<<"\n";
results<<"*****\n";
//Set max(-1), min(1) or without objective function (0);
sol1.setObjSense(1);
//STEP 3. OBTAINING OPTIMAL LINEAR SOLUTION AND INITIALIZATION OF CPLEX SOLVER
//Add the set of integer variables. For example the first stage
//variables
int setInt[nix];
for(i=0;i<nix;i++) setInt[i]=i;
for(i=0;i<nix;i++) sol1.setInteger(setInt[i]);
//OBTAINING A LINEAR SOLUTION
```

```

sol1.initialSolve();
if(sol1.isProvenPrimalInfeasible() ){results<<"The linear problem is infeasible "<<"\n";
goto 1969;}
if(!sol1.isProvenOptimal() ){results<<" The optimum is not found"<<"\n";
goto 1969;}
//zlp, since it is a minimization model
results<<"Optimal Linear solution:"<<-sol1.getObjValue()<<"\n";
results<<" First stage variables x** \n";
for (j=0;j<nix;j++) results<< sol1.getColSolution()[j]<< " ";
results<<"\n ";
results<<" Second stage variables y** \n";
for (iomega=0;iomega<nw;iomega++){
results<<"Scenario "<<iomega+1<<"\n";
for (j=0;j<niy;j++) results<<
sol1.getColSolution()[nix+iomega*niy+j]<< " ";
results<<"\n ";
}
// OBTAINING AN OPTIMAL MIXED-INTEGER SOLUTION WITH CPLEX
sol1.branchAndBound();
//WRITE DATA in mps file
sol1.writeMps("model-farmer");
if(sol1.getObjValue()>1e31){results<<"MIP Unbounded";
goto 1969;}
if(!sol1.isProvenOptimal()){results<<" The optimum is not found";
goto 1969;}
results<<"*****\n";
//-sol1.getColSolution()[j], since it is a minimization model
results<<"Optimal mixed-integer solution:"<<-sol1.getObjValue()<<"\n";
results<<" First stage variables x** \n";
for (j=0;j<nix;j++) results<< sol1.getColSolution()[j]<< " ";
results<<"\n ";
results<<" Second stage variables y** \n";
for (iomega=0;iomega<nw;iomega++){
results<<"Scenario "<<iomega+1<<"\n";
for (j=0;j<niy;j++) results<<
sol1.getColSolution()[nix+iomega*niy+j]<< " ";
results<<"\n ";
}
results<<"CPU output time COIN"<<CoinCpuTime()<<"\n";
tiempo1=CoinCpuTime();
results<<"*****\n";
results<<"TOTAL Time COIN: "<<tiempo1-tiempo01<<"\n";
1969: results.close();
return 0;}
//visual c++ version
#include "model-farmer.cpp"
#include "param-farmer.cpp"

```

Appendix B. File Makefile

```

# Copyright (C) 2006 International Business Machines and others.
# All Rights Reserved.
# This file is distributed under the Common Public License.
# $Id: Makefile.in 726 2006-04-17 04:16:00Z andreasw $
#####
# You can modify this example makefile to fit for your own program. #
# Usually, you only need to change the five CHANGEME entries below. #
#####
# To compile other examples, either changed the following line, or
# add the argument DRIVER=problem_name to make
#DRIVER =
DRIVER = farmer-cplex

```

```

# CHANGEME: This should be the name of your executable
EXE = $(DRIVER)
# CHANGEME: Here is the name of all object files corresponding to the source
#       code that you wrote in order to define the problem statement
#OBS = $(DRIVER).o
OBS = principal-farmer-cplex.o \
      model-farmer.o \
      param-farmer.o
SYSTEM = x86_sles10_4.1
# Directory with COIN header files
COININCDIR = /home/CoinAll/include
# Directory with COIN libraries
COINLIBDIR = /home/CoinAll/lib
# Directory with CPLEX
CPLEXDIR= /usr/ilog/CPLEX_Studio_AcademicResearch122/cplex
# Directory with CPLEX header files
CPLEXINCDIR = $(CPLEXDIR)/include/ilcplex
# Directory with CPLEX libraries
CPLEXLIBDIR=$(CPLEXDIR)/lib/$(SYSTEM)/static_pic/libcplex.a
CPLEXBINDIR=$(CPLEXDIR)/bin/$(SYSTEM)
# CHANGEME: Additional libraries
ADDLIBS =
# CHANGEME: Additional flags for compilation (e.g., include flags)
ADDINCFLAGS =
# CHANGEME: Directory to the sources for the (example) problem definition
# files
SRCDIR = .
#####
# Usually, you don't have to change anything below. Note that if you #
# change certain compiler options, you might have to recompile the #
# COIN package. #
#####
# C++ Compiler command
CXX = g++
# C++ Compiler options
CXXFLAGS = -O3 -fomit-frame-pointer -pipe -DNDEBUG -pedantic-errors -Wimplicit -Wparentheses -Wreturn-type
-Wcast-qual -Wall -Wpointer-arith -Wwrite-strings -Wconversion -m32 -O -fPIC -fexceptions -DIL_STD
# additional C++ Compiler options for linking
CXXLINKFLAGS = -Wl,-rpath -Wl,$(COINLIBDIR) -Wl,$(CPLEXLIBDIR)
# Libraries necessary to link with Clp and Cpx
LIBS = -L$(COINLIBDIR) -lCbcSolver -lCbc -lCgl -lOsiClp -lOsiCbc -lOsi -lClp -lCoinUtils -lOsiCpx
-L$(CPLEXBINDIR) -lcplex122 \
      -lm
# Necessary Include dirs (we use the CYGPATH_W variables to allow
# compilation with Windows compilers)
INCL = -I$(CYGPATH_W) $(COININCDIR) $(CPLEXINCDIR) $(ADDINCFLAGS)
# The following is necessary under cygwin, if native compilers are used
CYGPATH_W = echo
# Here we list all possible generated objects or executables to delete them
CLEANFILES = \
      principal-farmer-cplex.o pricipal-farmer-cplex \
      model-farmer.o model-farmer \
      param-farmer.o param-farmer
all: $(EXE)
.SUFFIXES: .cpp .c .o .obj

```



```

$(EXE): $(OBJS)
    bla=;\
    for file in $(OBJS); do bla="$$bla `$(CYGPATH_W) $$file`"; done; \
    $(CXX) $(CXXLINKFLAGS) $(CXXFLAGS) -o $@ $$bla $(ADDLIBS) $(LIBS)
clean:
    rm -rf $(CLEANFILES)
.cpp.o:
    $(CXX) $(CXXFLAGS) $(INCL) -c -o $@ `test -f '$<' || echo '$(SRCDIR)/'`$<
.cpp.obj:
    $(CXX) $(CXXFLAGS) $(INCL) -c -o $@ `if test -f '$<'; then $(CYGPATH_W) '$<'; else $(CYGPATH_W)
'$(SRCDIR)/$<; fi`
.c.o:
    $(CC) $(CFLAGS) $(INCL) -c -o $@ `test -f '$<' || echo '$(SRCDIR)/'`$<
clean_o:
    rm -rf $*.o
.c.obj:
    $(CC) $(CFLAGS) $(INCL) -c -o $@ `if test -f '$<'; then $(CYGPATH_W) '$<'; else $(CYGPATH_W)
'$(SRCDIR)/$<; fi`

```

Appendix C. File *resul-farmer.dat*

```

Farmer' Problem: OUTPUT
CPU time for loading data model 0
CPU input time COIN 0
Number of variables:21
Number of constraints:13
Number of nonzero elements:111
Matrix density:39.5714
*****
Optimal Linear solution:108390
First stage variables x**
170 80 250
Second stage variables y**
Scenario 1
0 0 310 48 6000 0
Scenario 2
0 0 225 0 5000 0
Scenario 3
0 48 140 0 4000 0
*****
Optimal mixed-integer solution:108390
First stage variables x**
170 80 250
Second stage variables y**
Scenario 1
0 0 310 48 6000 0
Scenario 2
0 0 225 0 5000 0
Scenario 3
0 48 140 0 4000 0
CPU output time COIN 0.004
*****
TOTAL Time COIN: 0.004

```

Appendix D. On using interactive CPLEX

In the installation of Windows, and in the directory `C:\ILOG\CPLEX_Studio_AcademicResearch122\cplex\bin\x86_win32` is the executable file `cplex.exe`. Copy it in the working directory `C:\coin-projects\farmer-cplex`.

In the Linux environment, and in the directory `/home/ILOG/CPLEX_Studio_AcademicResearch122/cplex/bin/x86_sles10_4.1` is also the executable `cplex`. Copy it in your working directory.

In both environments, and in the working directory, you must have the corresponding `.mps` file with the data of the example (you can obtain it from the partial execution of the code `principal-farmer-cplex.cpp`). It is named `model-farmer.mps`.

If you execute this program (in Windows, clicking two times over the file `cplex.exe`, and in Linux typing in the prompt `S ./cplex`). In both cases you will see the prompt `CPLEX>`

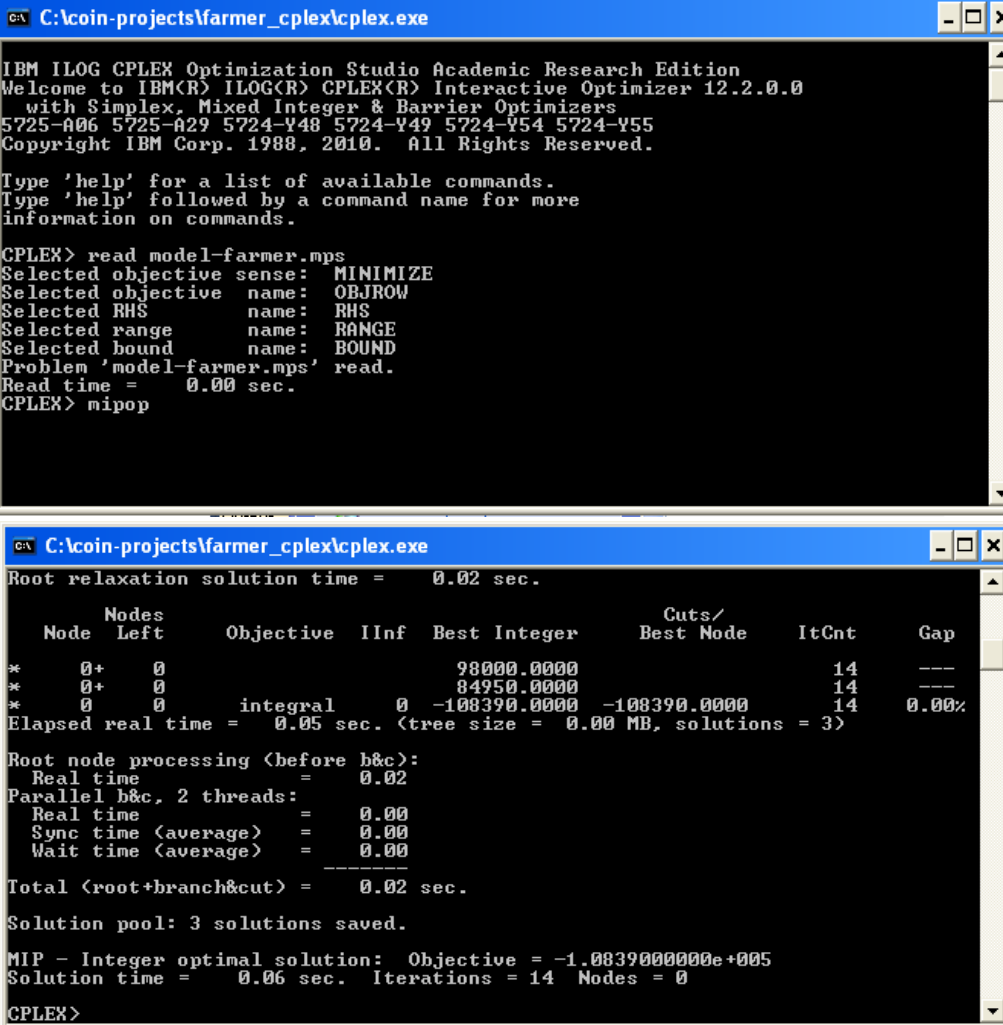
Now, you must type:

```
CPLEX> read model-farmer.mps
```

And then,

```
CPLEX> mipop
```

You will solve the minimization farmer' problem whose coefficients are in the file `model-farmer.mps`, and will see the following screens,



```
C:\coin-projects\farmer_cplex\cplex.exe
IBM ILOG CPLEX Optimization Studio Academic Research Edition
Welcome to IBM(R) ILOG(R) CPLEX(R) Interactive Optimizer 12.2.0.0
with Simplex, Mixed Integer & Barrier Optimizers
5725-A06 5725-A29 5724-Y48 5724-Y49 5724-Y54 5724-Y55
Copyright IBM Corp. 1988, 2010. All Rights Reserved.

Type 'help' for a list of available commands.
Type 'help' followed by a command name for more
information on commands.

CPLEX> read model-farmer.mps
Selected objective sense: MINIMIZE
Selected objective name: OBJROW
Selected RHS name: RHS
Selected range name: RANGE
Selected bound name: BOUND
Problem 'model-farmer.mps' read.
Read time = 0.00 sec.
CPLEX> mipop

Root relaxation solution time = 0.02 sec.

   Nodes
   Node Left   Objective IInf Best Integer   Cuts/
                                     Best Node   ItCnt   Gap
*    0+    0          98000.0000          14   ---
*    0+    0          84950.0000          14   ---
*    0    0   integral    0 -108390.0000 -108390.0000  14  0.00%
Elapsed real time = 0.05 sec. (tree size = 0.00 MB, solutions = 3)

Root node processing (before b&c):
  Real time = 0.02
Parallel b&c, 2 threads:
  Real time = 0.00
  Sync time (average) = 0.00
  Wait time (average) = 0.00
-----
Total (root+branch&cut) = 0.02 sec.

Solution pool: 3 solutions saved.

MIP - Integer optimal solution: Objective = -1.0839000000e+005
Solution time = 0.06 sec. Iterations = 14 Nodes = 0

CPLEX>
```

To change the optimization direction to maximizing, set CPLEX>change sense obj max

The input file "model-farmer.mps" is

NAME	OsiDefaultName		
ROWS			
N	obj		
L	c1		
L	c2		
L	c3		
G	c4		
G	c5		
L	c6		
L	c7		
G	c8		
G	c9		
L	c10		
L	c11		
G	c12		
G	c13		
COLUMNS			
MARK0000	'MARKER'		'INTORG'
x1	obj		150
x1	c1		1
x1	c2		-3
x1	c6		-2.5
x1	c10		-2
x2	obj		230
x2	c1		1
x2	c3		-3.6
x2	c7		-3
x2	c11		-2.4
x3	obj		260
x3	c1		1
x3	c4		-24
x3	c8		-20
x3	c12		-16
MARK0001	'MARKER'		'INTEND'
x4	obj	79.33333333333333	
x4	c2		-1
x5	obj		70
x5	c3		-1
x6	obj	-56.66666666666667	
x6	c2		1
x7	obj		-50
x7	c3		1
x8	obj		-12
x8	c4		1
x8	c5		1
x9	obj	-3.333333333333333	
x9	c4		1
x10	obj	79.33333333333333	
x10	c6		-1
x11	obj		70
x11	c7		-1
x12	obj	-56.66666666666667	
x12	c6		1
x13	obj		-50
x13	c7		1
x14	obj		-12
x14	c8		1
x14	c9		1
x15	obj	-3.333333333333333	
x15	c8		1
x16	obj	79.33333333333333	
x16	c10		-1
x17	obj		70
x17	c11		-1
x18	obj	-56.66666666666667	
x18	c10		1
x19	obj		-50
x19	c11		1

```

x20      obj          -12
x20      c12           1
x20      c13           1
x21      obj      -3.333333333333333
x21      c12           1
RHS
rhs      c1           500
rhs      c2          -200
rhs      c3          -240
rhs      c4         -12000
rhs      c6          -200
rhs      c7          -240
rhs      c8         -12000
rhs      c10         -200
rhs      c11         -240
rhs      c12        -12000
RANGES
rng      c4           12000
rng      c5            6000
rng      c8           12000
rng      c9            6000
rng      c12          12000
rng      c13            6000
BOUNDS
LI bnd   x1            0
LI bnd   x2            0
LI bnd   x3            0
ENDATA

```

Gratefulnesses

This research has been partially supported by the projects ECO2008-00777 ECON from the Ministry of Education and Science, and Grupo de Investigación IT-347-10 from the Basque Government, Spain.

References

- [1] J.R. Birge and F.V. Louveaux. *Introduction to Stochastic Programming*. Springer, 1997.
- [2] IBM ILOG CPLEX <http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/>, 2011.
- [3] INFORMS. COIN-OR: *Computational Infrastructure for Operations Research*. <http://www.coin-or.org>, 2011.
- [4] R. Laugee-Heimer. The *COmmon INfrastructure for Operations Research*, IBM Journal of Research and Development, 47(1): 55-66, 2003.
- [5] Microsoft. *Visual C++ Express Edition 2010*. <http://www.microsoft.com>.
- [6] G. Pérez and M.A. Garín. On downloading and using COIN-OR for solving linear/integer optimization problems. Working paper series Biltoki. DT.2010.05. Edited by Dpto. Economía Aplicada III. Universidad del País Vasco, UPV/EHU (Spain), 2010. <http://econpapers.repec.org/paper/ehubiltok/201005.htm>.