DOCUMENTOS DE TRABAJO

BILTOKI

D.T. 2010.05

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

Gloria Pérez Sainz de Rozas y María Araceli Garín Martín

eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

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

Documento de Trabajo BILTOKI DT2010.05
Editado por el Departamento de Economía Aplicada III (Econometría y Estadística)
de la Universidad del País Vasco.

Depósito Legal No.: BI-1424/2010
ISSN: 1134-8984
URN: RePEc:ehu:biltok:201005

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

Gloria Pérez[1] and M.Araceli Garín[2]

[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 help to use the open source software for optimization COIN-OR. In particular, we describe how to download, install and use the corresponding solvers under Windows and Linux operating systems. We will use an example taken from the literature, with the corresponding source code written in C++, to describe the whole process of editing, compiling and running the executable, to solve this optimization problem by using this software.

1. Introduction

COIN-OR, which stands for Computational Infraestructure for Operations Research, is a collection of open source software for optimization, see http://www.coin-or.org . The code is written in C++, and it means that can 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. 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 libraries from Clp (LP solvers), Cbc (MIP solvers) and Cgl (Cut generators) and some others libraries with general and interface applications to generate the executable for solving mixed integer problems.

In order to illustrate the procedure step by step, we will use the farmer's problem taken from Birge and Loveaux (1997). We will describe the whole procedure under Windows and UNIX-like operating systems. The reminder 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 installation of COIN-OR software under Windows system. Section 4, presents the details to create a Visual C++ project, in order to generate an executable (solution) and running it, to solve an optimization problem. Section 5, describes the downloading process and basic installation of COIN-OR software under Unix-like systems. Section 6, gives the main steps to link your own code with COIN-OR software and running the executeble under Linux. In the Appendix A, we summarize the source code written in C++, for the example. In the Appendix B, we add the Makefile for compiling undex Linux, and finally, in Appendix C, we present the output file to check the results of farmer' problem.

2. Illustrative case

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

$$
\begin{aligned}
min\; 150x_1 + 230x_2 + 260x_3 &- \frac{1}{3}(170w_{11} - 238y_{11} + 150w_{21} - 210y_{21} + 36w_{31} + 10w_{41})\\
&- \frac{1}{3}(170w_{12} - 238y_{12} + 150w_{22} - 210y_{22} + 36w_{32} + 10w_{42})\\
&- \frac{1}{3}(170w_{13} - 238y_{13} + 150w_{23} - 210y_{23} + 36w_{33} + 10w_{43})
\end{aligned}
$$

$$
\begin{aligned}
&x_1 + x_2 + x_3 \le 500,\\
&3x_1 + y_{11} - w_{11} \ge 200,\\
&2.5x_1 + y_{12} - w_{12} \ge 200,\\
&2x_1 + y_{13} - w_{13} \ge 200,\\
&3.6x_2 + y_{21} - w_{21} \ge 240,\\
&3x_2 + y_{22} - w_{22} \ge 240,\\
&2.4x_2 + y_{23} - w_{23} \ge 240,
\end{aligned}
$$

$$
\begin{aligned}
&w_{31} + w_{41} \le 24x_3, \\
&w_{32} + w_{42} \le 20x_3, \\
&w_{33} + w_{43} \le 16x_3, \\
&w_{31} \le 6000, \\
&w_{32} \le 6000, \\
&w_{33} \le 6000, \\
&x_i, y_{j\omega}, w_{i\omega} \ge 0, \quad \forall i, j, \omega
\end{aligned}
$$

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, $w_{i\omega}$ $i = 1, \cdots 4$ 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 1/3, the farmer' problem is reading 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.

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,…,4 and $y_{j\omega}$, j=1,2, are second stage variables, which represent decisions about sales and purchases, that depend on the yiels. These decisions depend also of a scenario index ω, with ω=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 1/3, the farmer' problem reads as given above. The optimal solution value is 108390, as you can check in the result 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,

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

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 is *principal-farmer.cpp*. In this program is also described how to solve the linear model if you can read the data 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. This is the file *pm.h.*

3. Basic installation under WINDOWS-like systems

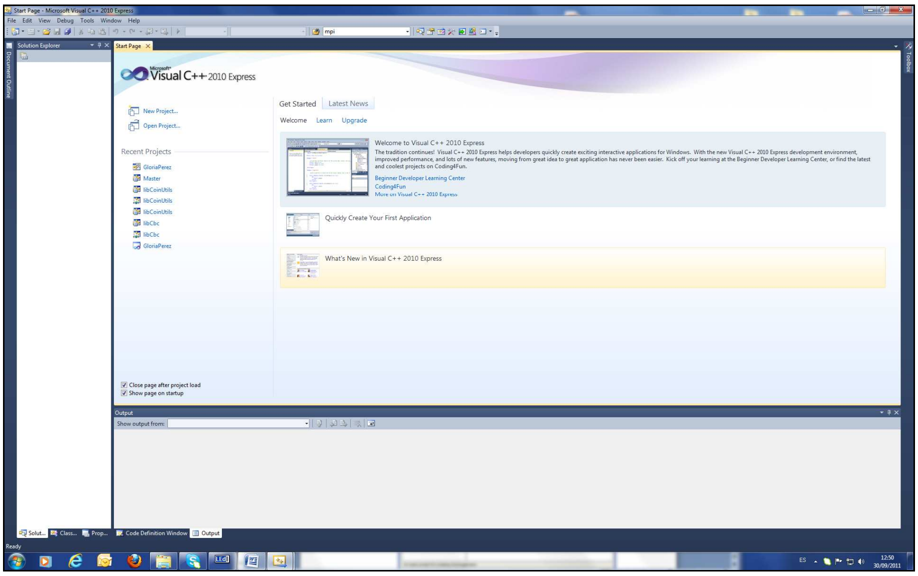
The first step is to download the code. For doing it, you must click on Download/Use in the left hand side of the home page http://www.coin-or.org. Then in the second Section titled Source Code, you must click on here to download the source code for the latest stable release. You can observe an index 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 and select CoinAll-1.4.0.zip.

Alternatively you can go directly to the corresponding home page http://www.coin-or.org/download/source/CoinAll/CoinAll-1.4.0.zip. After downloading the tarball you must extract the code into a new subdirectory, C:\CoinAll.

You need a windows C++ compiler, and you can download the Visual C++ 2008 Express Edition (30/90 days free) from http://www.microsoft.com/Express/VC. After downoading and save the file vcstup.exe in your computer, you must execute it.

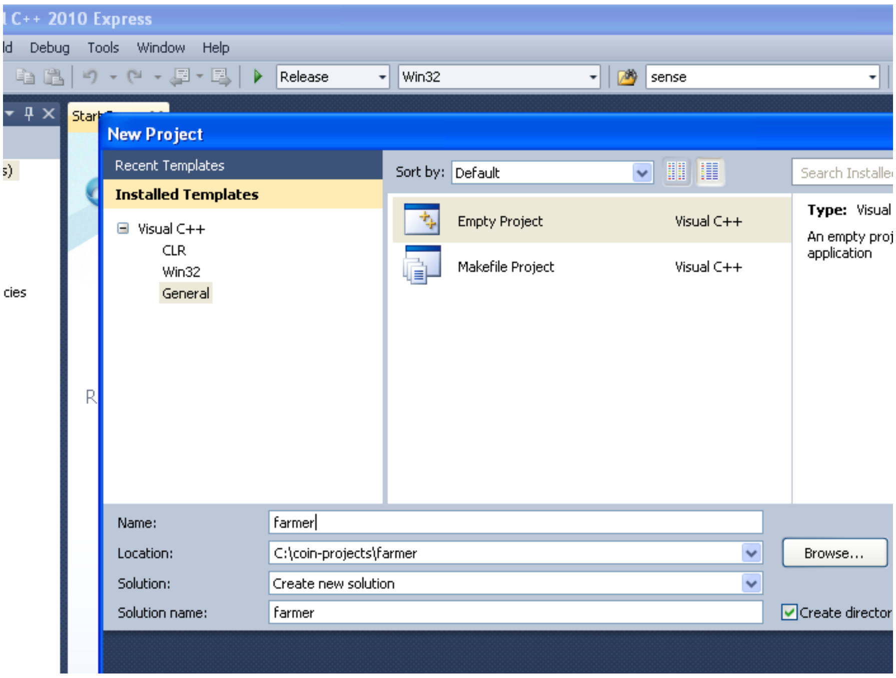
You must choose the option "registrate on line", and with a hotmail account and its password you can get the fourteen digits code to registrate free use the product.

After executing the file vcstup.exe, several requirements are downloaded and installed in your computer. Now you can access to the compiler start page, that might look like this:



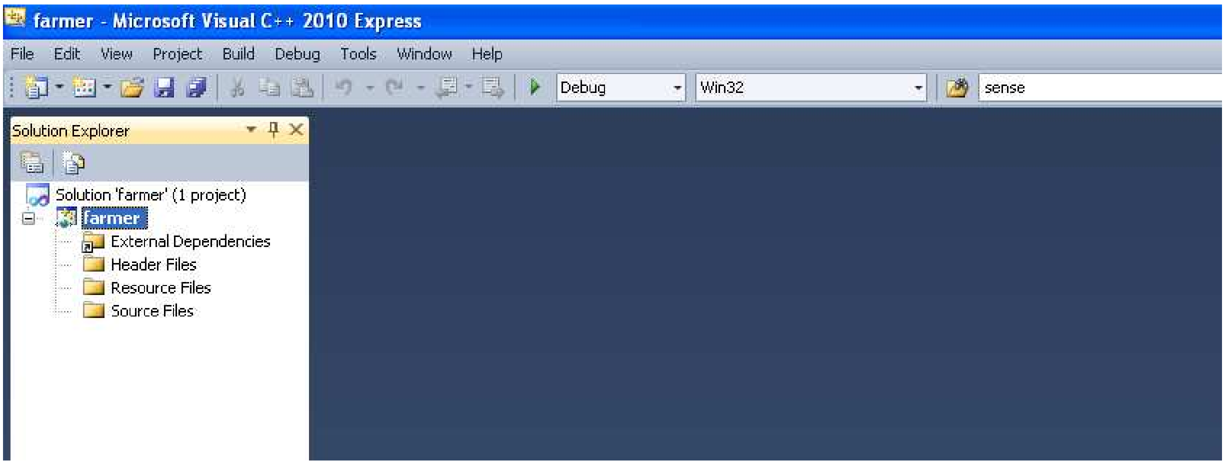
4. Linking your code with COIN and running the executable under Windows

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, where your source code is.
From the desktop, click on Visual C++ to open it. You must select New Project in the File options menu. In this screen, you must choose: General, Empty Project, a name for your project, for example “farmer”, enter the working directory and a name if you want to create a new subdirectory for the solution. With all these changes, the screen look likes this:



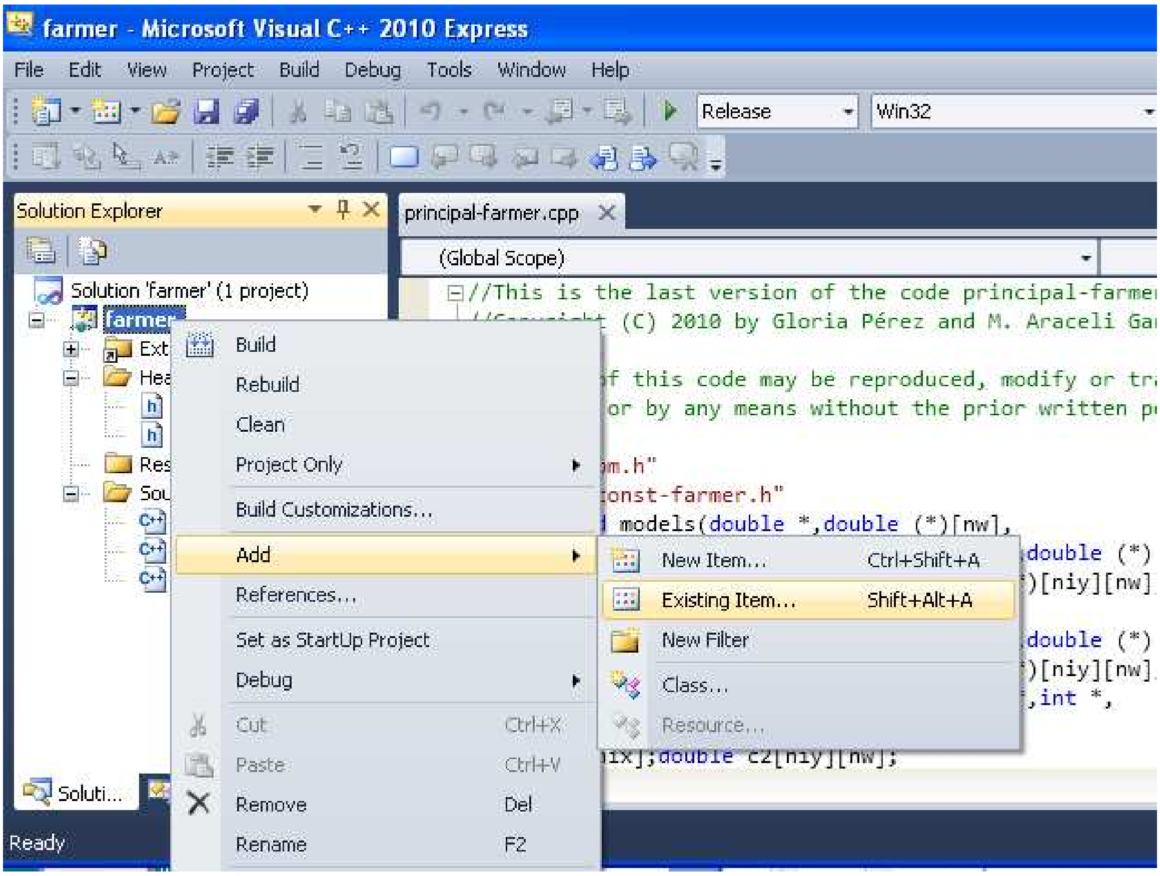
In this working directory you have to put your own code, files .cpp and header files .h

If you click on OK, the project *farmer* is created, and look likes this:

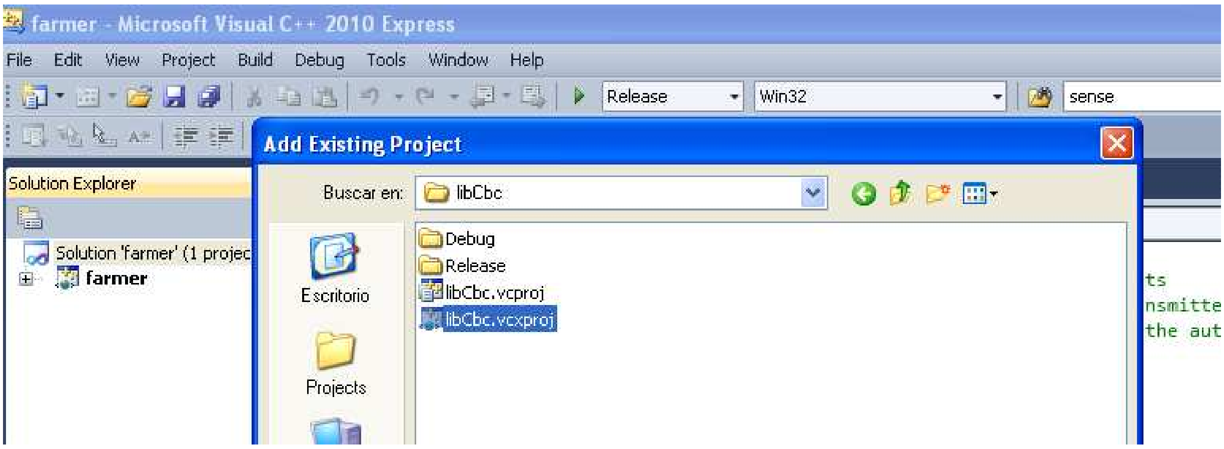


Also you must change the Solution Configurations in the front of the screen, from Debug to Release.

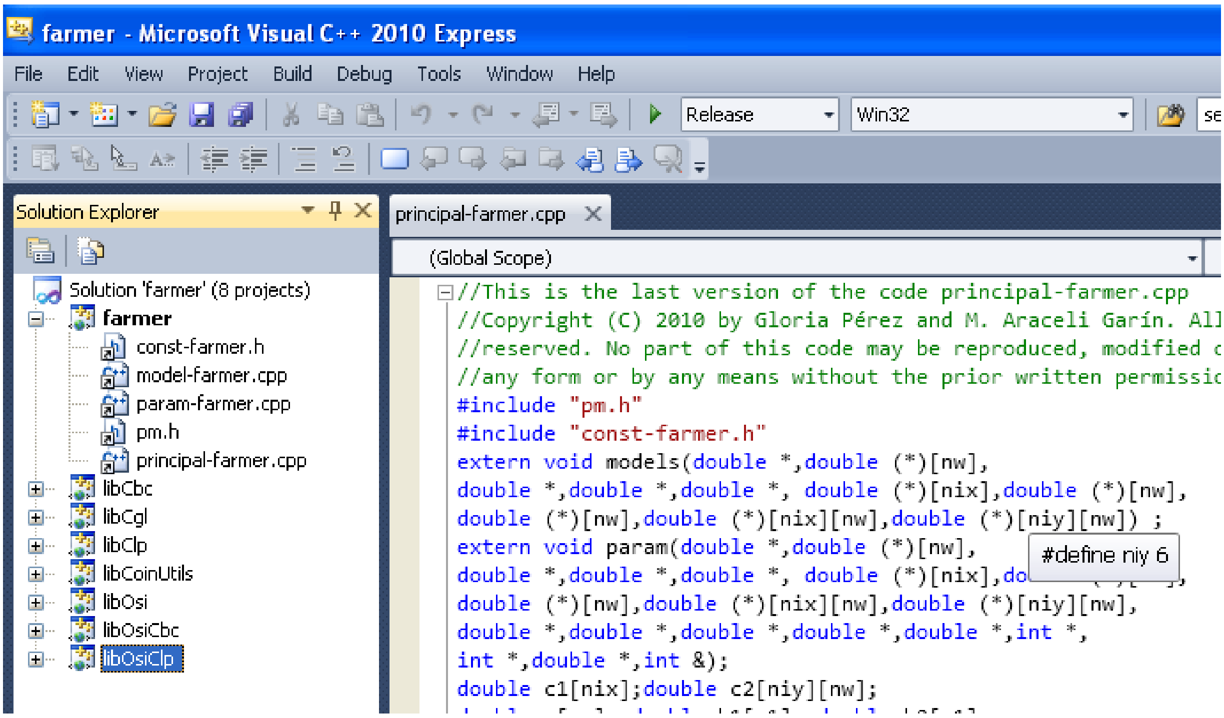
You must click on File and open the file *principal-farmer.cpp* from the Visual C++ compiler. You can also do it, by adding your own source files of code (.cpp) and the headers (.h) as existing items. To do it, you must click on the right button over Source Files, and over Header Files. Then, you can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



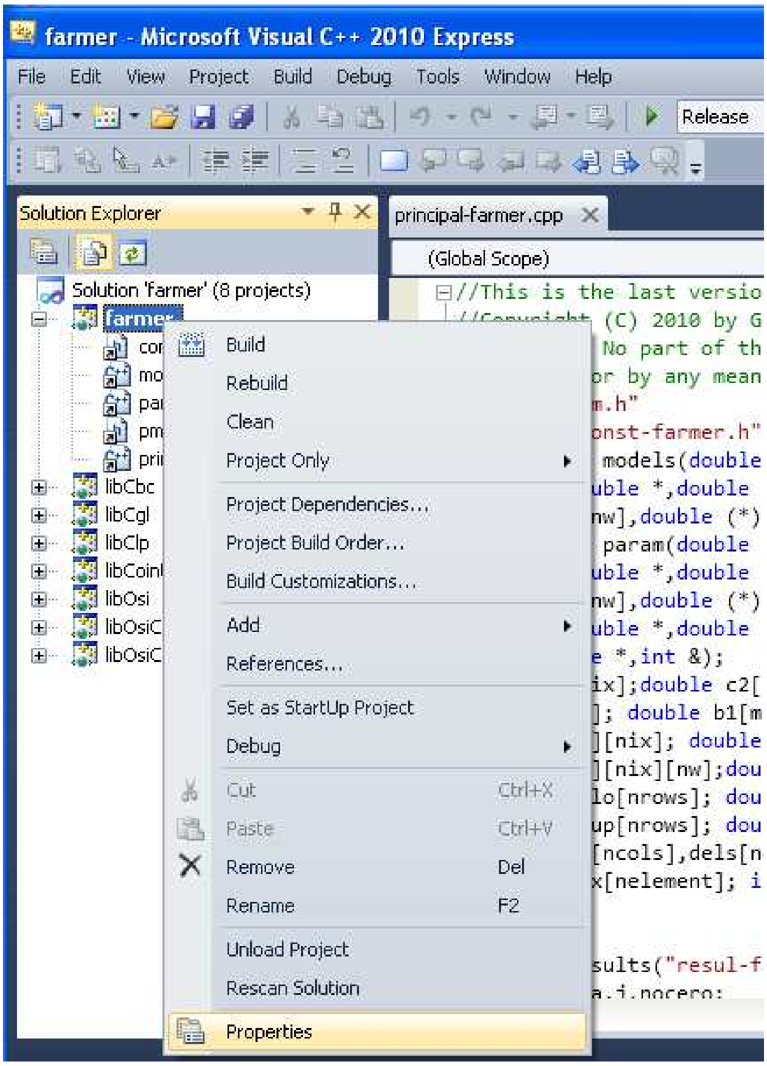
After doing it, you must click again in File and add as existing projects seven COIN projects. You must look for the path to the directory where are each of them. Then click two times on the file with extension .vcproj to open and add it.



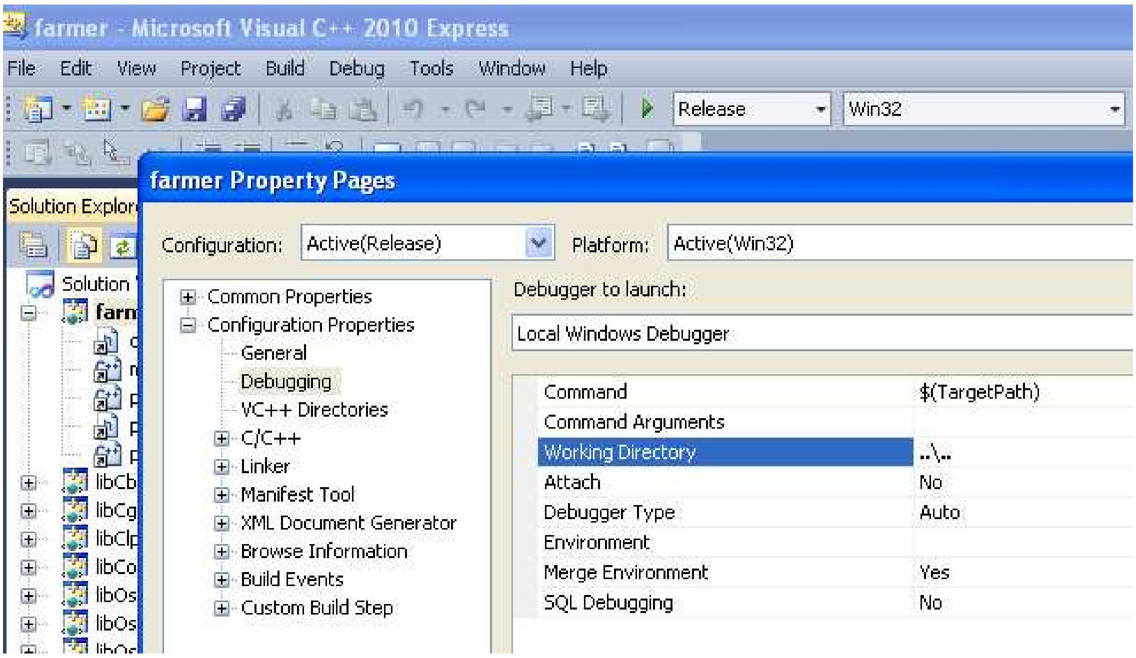
The seven paths and projects are:
C:\CoinAll\Cbc\MSVisualStudio\v9\libCbc\libCbc.vcproj
C:\CoinAll\Cgl\MSVisualStudio\v9\libCgl\libCgl.vcproj
C:\CoinAll\Clp\MSVisualStudio\v9\libClp\libClp.vcproj
C:\CoinAll\CoinUtils\MSVisualStudio\v9\libCoinUtils\libCoinUtils.vcproj
C:\CoinAll\Osi\MSVisualStudio\v9\libOsi\libOsi.vcproj
C:\CoinAll\Osi\MSVisualStudio\v9\libOsiCbc\libOsiCbc.vcproj
C:\CoinAll\Osi\MSVisualStudio\v9\libOsiClp\libOsiClp.vcproj
After doing it, the screen look likes this:



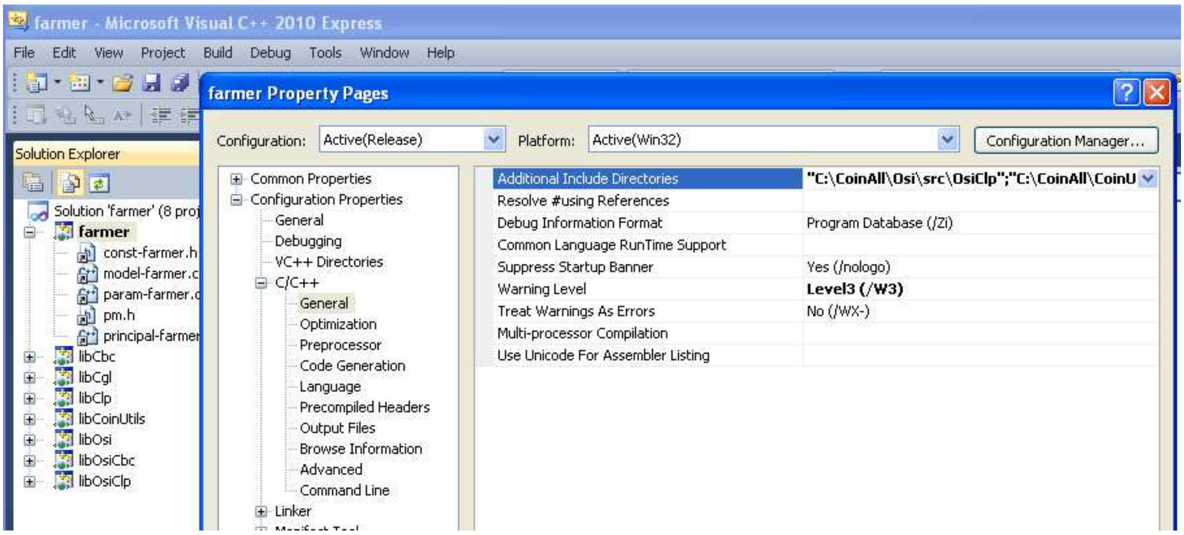
Finally, you must modify the properties of your project. To do it, click on the right button beside the boldface name of the project, **farmer**, and click on the last item, Properties.



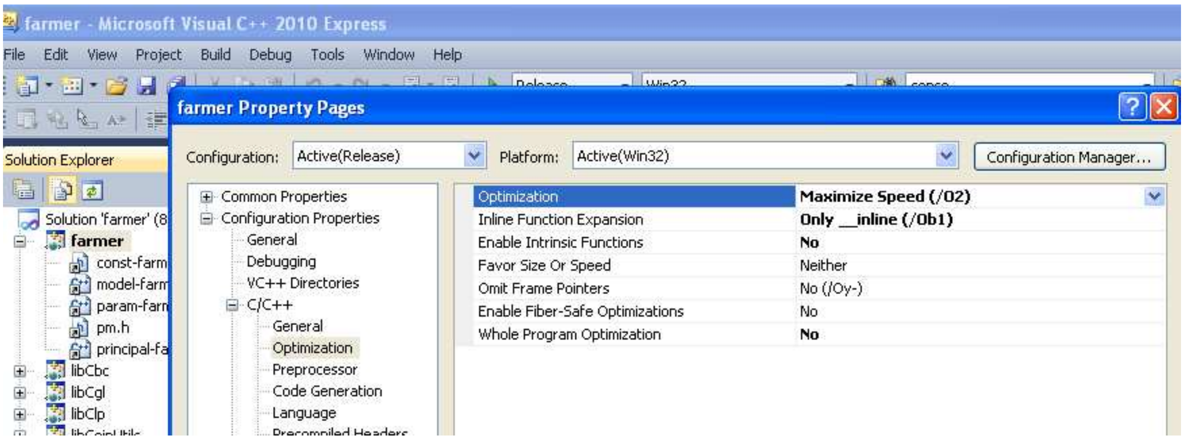
In the Debugging options, you must establish the working directory, C:\coin-projects\farmer, , and click on Aplicar button.



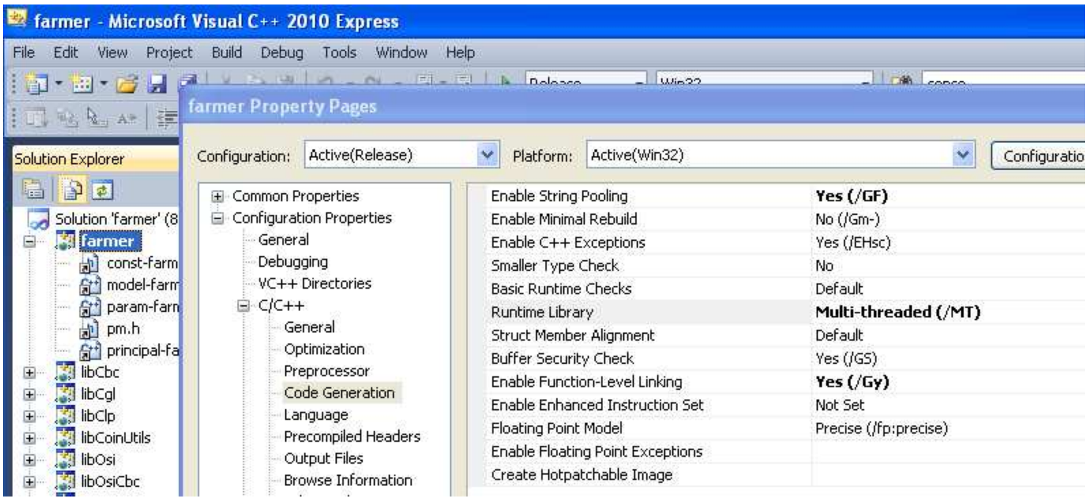
And you must write in this way, the following additional include directories:
"C:\CoinAll\Cgl\src\CglProbing";"C:\CoinAll\Cgl\src\CglGomory";
"C:\CoinAll\Cgl\src\CglClique";"C:\CoinAll\Cgl\src\CglKnapsackCover";
"C:\CoinAll\Cgl\src";"C:\CoinAll\Clp\src";
"C:\CoinAll\Cbc\src";"C:\CoinAll\Osi\src";
"C:\CoinAll\BuildTools\headers";"C:\CoinAll\Osi\src\OsiCbc";
"C:\CoinAll\Osi\src\OsiClp";"C:\CoinAll\CoinUtils\src";
These are the directories with the included files that appear in the header file *pm.h*



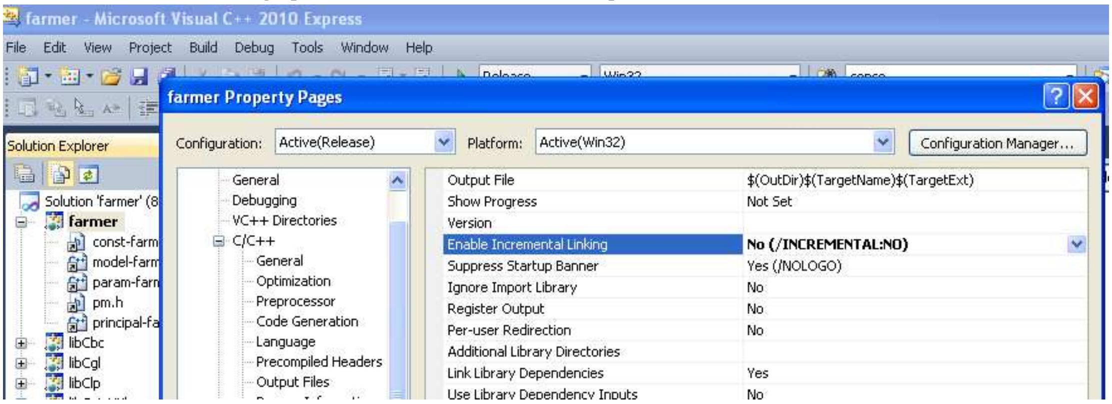
In the Optimization options, set as follows and click on Aplicar button



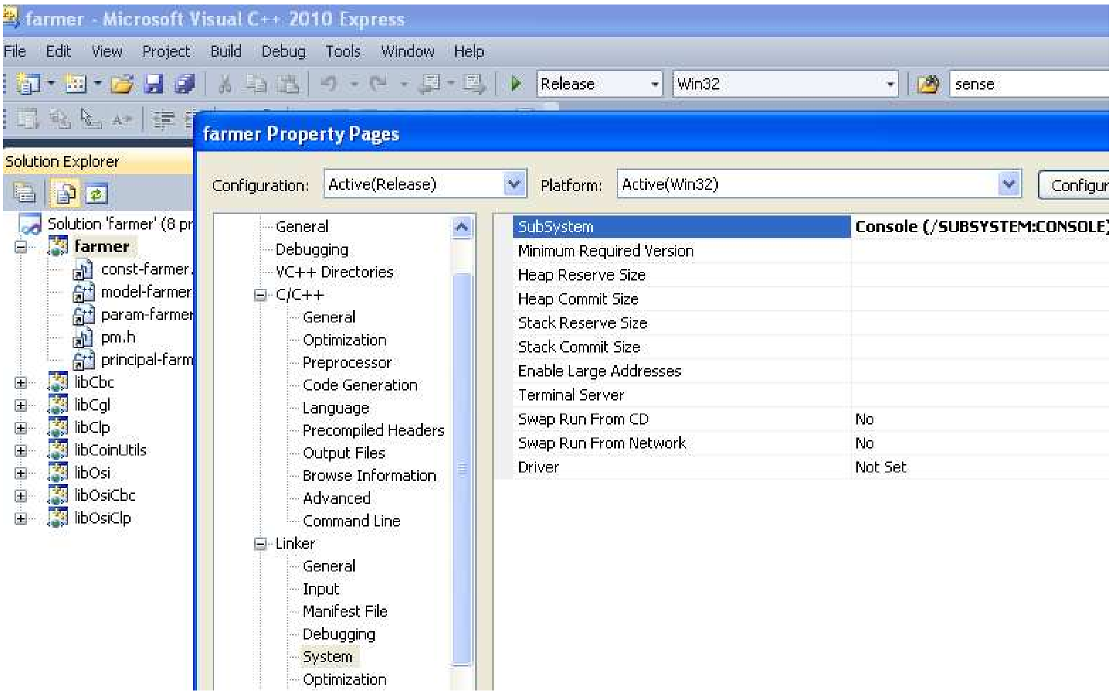
In Code Generation, set as follows, and click on Aplicar button.



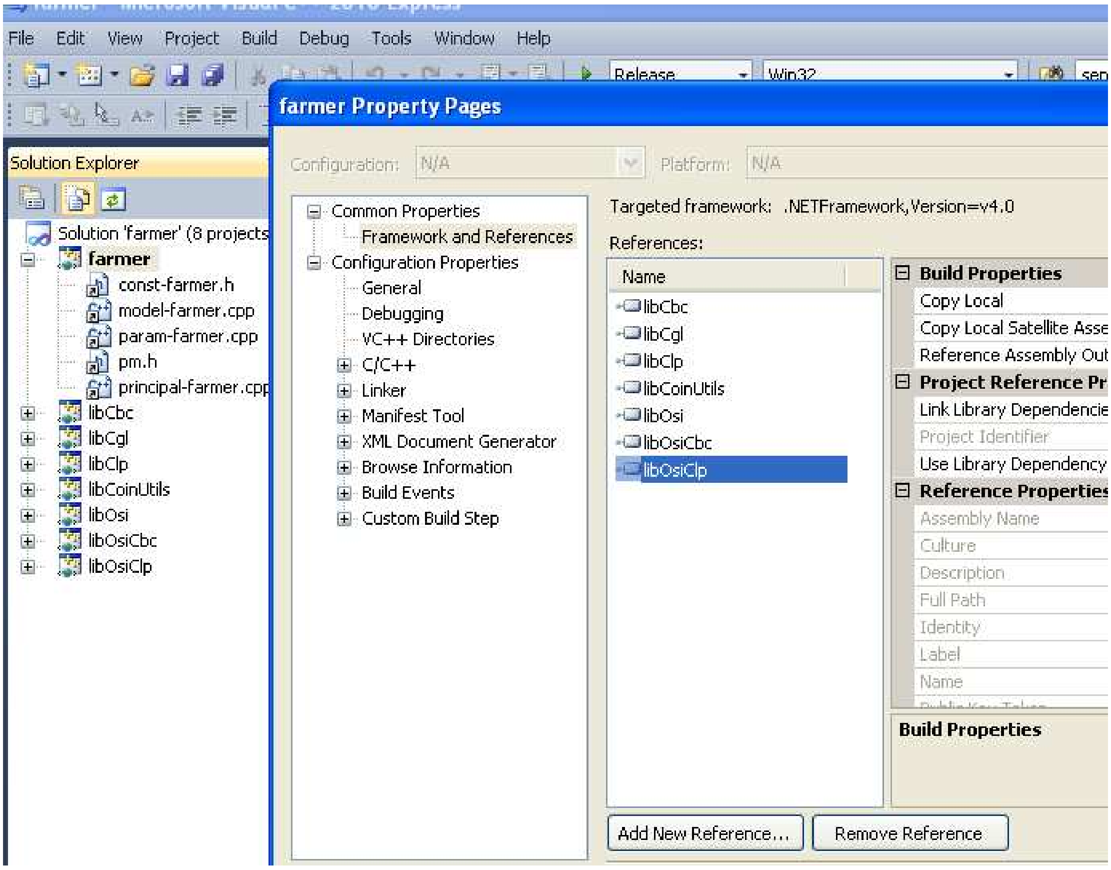
In Linker, set the following options in General, and click on Aplicar button.



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

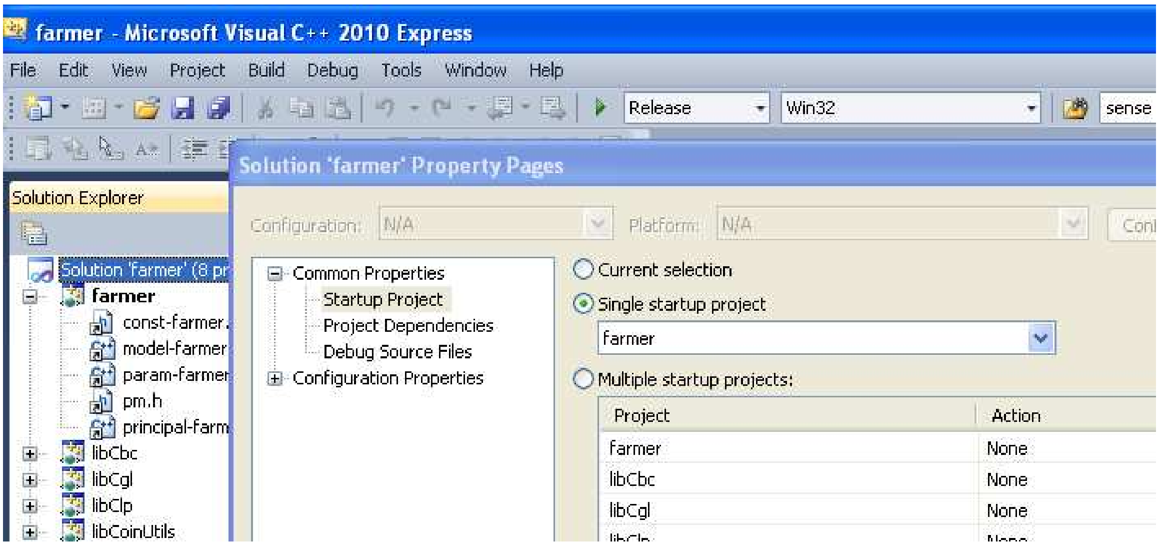


In the left menu, in Common Properties you must add seven new references, clicking on the Add New Reference button, and select the seven projects: libCbc, libCgl, libClp, libCoinUtils, libOsi, libOsiCbc and libOsiClp.

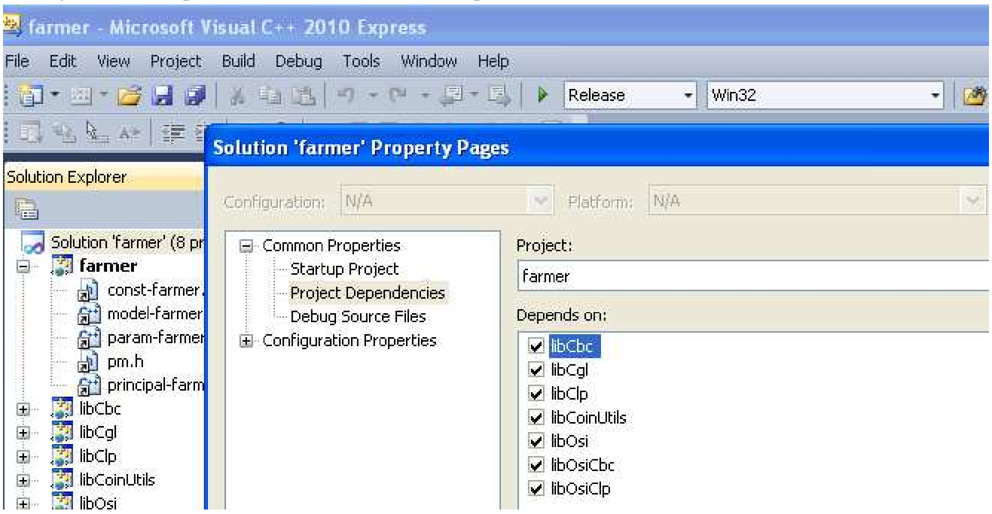


Finally, click on Aplicar button and on Aceptar button to go to the start project screen.

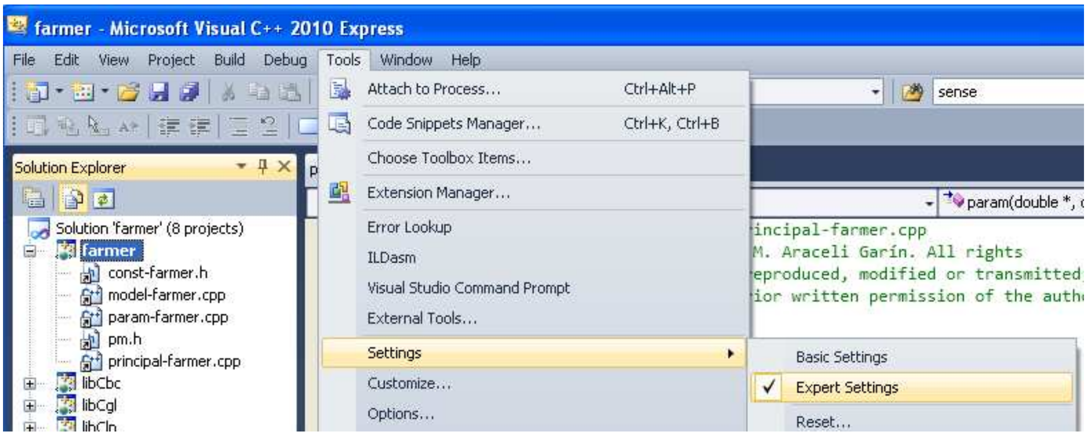
Now you must modify some properties of your solution (executable). To do it, click on the right button over the icon beside Solution `farmer´ (8 projects), and select the last option, Properties. In the left screen, select Startup Project in the Common Properties, and in the right screen click on Single startup project to select *farmer* project. After this, click on Aplicar button.



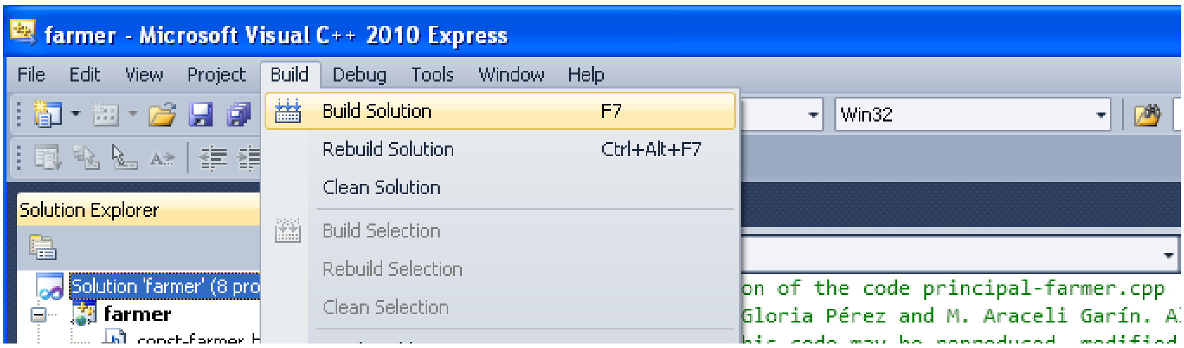
Then, to set the Project Dependences, select in the right hand screen farmer as project, and click on all the projects given below. Finally, click on Aplicar button and then, on Aceptar button.



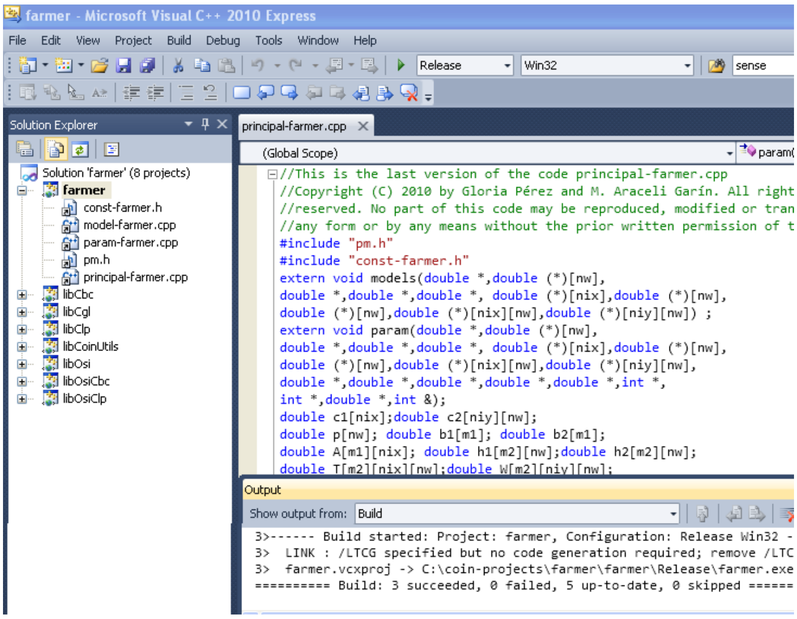
Before running the solution (executable), select in the Tools window, Settings and Expert Settings.



Now you can compile and link your code, i.e. you can build the solution, *farmer.sln*. To do it, in the start screen, click on Build and then, on Build Solution.

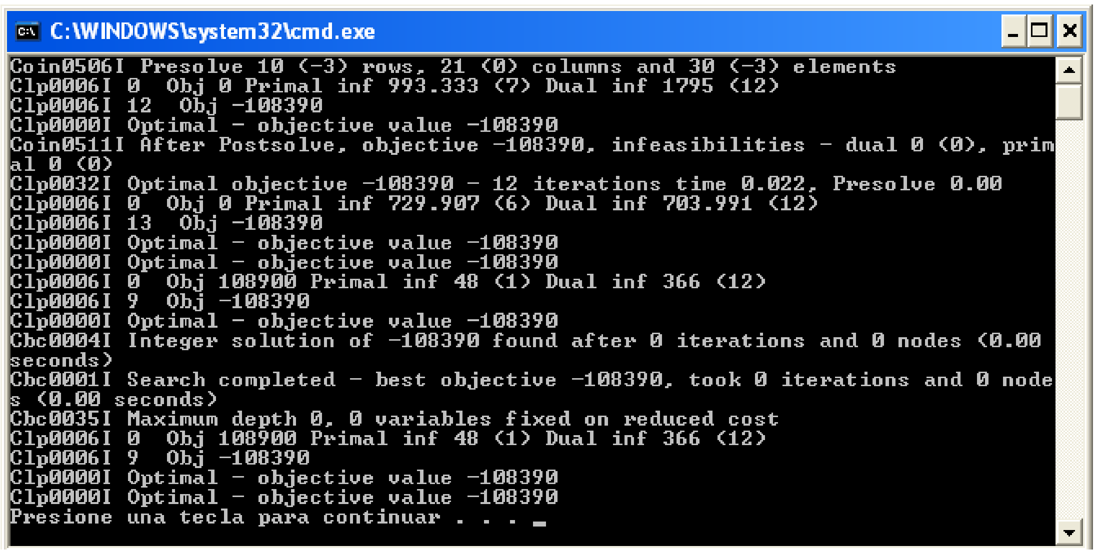


If all went fine, you obtain 0 errors, and the Solution (executable) has been built. This information appears in the bottom of the screen. The screen look likes this:



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 files *resul-farmer.dat, output-models.dat and output-param.dat* have been generated in your working directory.



5. Basic installation under UNIX-like systems

Again, the first step is to download the code of COIN-OR. 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 tittled 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 Unix-like enviroments 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, with

$ tar xzvf CoinAll-1.4.0.tgz

Also you can create a subdirectory, for example named CoinAll and then, 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 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.

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

Assume that you have downloaded the COIN sources in the directory CoinAll and you have run configure, make and make install obtaining a set of packages, each one in a subdirectory, libraries in subdirectory /lib and include files in subdirectory /include.

For most COIN 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 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.cpp* and *pm.h.* In order to modify this Makefile to compile your own code, you only have to change some things. Edit the Makefile, and 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". 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

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

$ ./farmer

Again, if all went fine you will see at the end of the output,

```
Cbc0035I Maximum depth 0, 0 variables fixed on reduced cost
Clp0006I 0  Obj 0 Primal inf 729.907 (6) Dual inf 703.991 (12)
Clp0006I 15  Obj -108390
Clp0000I Optimal - objective value -108390
Clp0000I Optimal - objective value -108390
```

Also in your directory, now there exists a new file, *resul-farmer.dat*, with the output of the execution, see Appendix C. The files *output-models.dat* and *output-param.dat* have also been generated in your working directory.

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.

7. Appendix A

7.1 File *const-farmer.h*

```
//This is the last version of the code const-farmer.h
//Copyright (C) 2010 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 continuos variables
#define nix 3
//Number of second stage continuos 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
```

7.2 File *pm.h*

```
//This is the last version of the code pm.h
//Copyright (C) 2010 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"
```

7.3 File *model-farmer.cpp*

```
//This is the last version of the code model-farmer.cpp
//Copyright (C) 2010 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 weigths
     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.;
}
```

7.4 File *param-farmer.cpp*

```
//This is the last version of the code param-farmer.cpp
//Copyright (C) 2010 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
}
```

7.5 File *principal-farmer.cpp*

```
//This is the last version of the code principal-farmer.cpp
//Copyright (C) 2010 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 COIN-OR

//without using interface (OSIClpSolverinterface)
//    ClpSimplex *sol1; sol1=new ClpSimplex;
//or alternatively, by using interface

        OsiClpSolverInterface 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 time: input 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);

//WRITE DATA in mps file
            sol1.writeMps("model-farmer");

//STEP 3. OBTAINING OPTIMAL LINEAR SOLUTION  AND INITIALIZATION OF CBC //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]);
            CbcModel pm1(sol1);

//OBTAINING A LINEAR SOLUTION
            sol1.initialSolve();
            if(sol1.isProvenPrimalInfeasible() ){results<<"The linear problem is infeasible "<<"\n";
            goto l969;}
            if(!sol1.isProvenOptimal() ){results<<" The optimum is not found"<<"\n";
```

```
            goto l969;}
        //-zlp, since it  is  a minimization model
results<<"Optimal Linear solution with COIN:"<<-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
            pm1.branchAndBound();
if(pm1.getBestPossibleObjValue()>1e31){results<<"MIP Unbounded";
                        goto l969;}
if(!pm1.isProvenOptimal()){results<<"  The optimum is not found";
                        goto l969;}

results<<"*****************************************\n";
//-pm1.getColSolution()[j], since it  is  a minimization model
results<<"Optimal mixed-integer solution with COIN:"<<pm1.getObjValue()<<"\n";
results<<" First stage variables x** \n";
            for (j=0;j<nix;j++) results<<  pm1.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<<  pm1.getColSolution()[nix+iomega*niy+j]<< " ";
                        results<<"\n  ";
                        }
             results<<"CPU time: output COIN "<<CoinCpuTime()<<"\n";
             tiempo1=CoinCpuTime();       results<<"******************************\n";
             results<<"TOTAL Time COIN: "<<tiempo1-tiempo01<<"\n";
l969:    results.close();
               return 0;}

//visual c++ version
#include "model-farmer.cpp"
#include "param-farmer.cpp"
```

8. **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 = farmer
# 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
#OBJS =  $(DRIVER).o
OBJS= principal-farmer.o \
```

```
model-farmer.o\
param-farmer.o\
# CHANGEME: Additional libraries
ADDLIBS = -D.
# CHANGEME: Additional flags for compilation (e.g., include flags)
ADDINCFLAGS =-I.
# CHANGEME: Directory to the sources for the (example) problem definition
# files
SRCDIR = /home/coin-projects.
##################################################################
#  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
# additional C++ Compiler options for linking
CXXLINKFLAGS =  -Wl,--rpath -Wl,/home/CoinAll/lib
# C Compiler command
CC = gcc
# C Compiler options
CFLAGS = -O3 -fomit-frame-pointer -pipe -DNDEBUG -pedantic-errors -Wimplicit -Wparentheses -Wsequence-point -Wreturn-type -Wcast-qual -Wall
# Directory with COIN header files
COININCDIR = /home/CoinAll/include
# Directory with COIN libraries
COINLIBDIR = /home/CoinAll/lib
# Libraries necessary to link with Clp
LIBS = -L$(COINLIBDIR) -lCbc -lCgl -lOsiClp -lOsi -lClp -lCoinUtils \
-lm  \
`cat $(COINLIBDIR)/cgl_addlibs.txt` \
`cat $(COINLIBDIR)/clp_addlibs.txt` \
`cat $(COINLIBDIR)/coinutils_addlibs.txt`
#LIBS = -L$(COINLIBDIR) -lClp -lCoinUtils \
#-lm  `cat $(COINLIBDIR)/coinutils_addlibs.txt`
# Necessary Include dirs (we use the CYGPATH_W variables to allow
# compilation with Windows compilers)
INCL =  -I`$(CYGPATH_W) $(COININCDIR)` $(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 = \
addBits.o addBits \
addColumns.o addColumns \
addRows.o addRows \
decompose.o decompose \
defaults.o defaults \
driver2.o driver2 \
driver.o driver \
driverC.o driverC \
dualCuts.o dualCuts \
ekk.o ekk \
ekk_interface.o ekk_interface \
hello.o hello \
```

```
makeDual.o makeDual \
minimum.o minimum \
network.o network \
piece.o piece \
rowColumn.o rowColumn \
sprint2.o sprint2 \
sprint.o sprint \
testBarrier.o testBarrier \
testBasis.o testBasis \
testGub2.o testGub2 \
testGub.o testGub \
testQP.o testQP \
useVolume.o useVolume
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 '$(SRCDIR)/'`$<
.c.obj:
$(CC) $(CFLAGS) $(INCL) -c -o $@ `if test -f '$<'; then $(CYGPATH_W) '$<'; else $(CYGPATH_W) '$(SRCDIR)/$<'; fi`
```

9. Appendix C. File *resul-farmer.dat*

```
Farmer' Problem: OUTPUT
CPU time for loading data model 0
CPU time: input COIN 0
Number of variables:21
Number of constraints:13
Number of nonzero elements:111
Matrix density:39.5714
******************************************
Optimal Linear solution with COIN: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 with COIN: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 time: output COIN 0.004
*******************************
TOTAL Time COIN: 0.004
```


10. 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.

11. References

[1] J.R. Birge and F.V. Louveaux. *Introduction to Stochastic Programming.* Springer, 1997.

[2] INFORMS. COIN-*OR: Computational Infrastructure for Operations Research*. http://www.coin-or.org, 2008.

[3] R. Laugee-Heimer. The *Common INfrastructure for Operations Research*, IBM Journal of Research and Development, 47(1): 55-66, 2003.

[4] Microsoft. *Visual C++ Express Edition*. http://www.microsoft.com, 2008.