FICO™ Xpress Optimization Suite

Guide for evaluators

Last update 26 May, 2009

FICO™

FICO™ Xpress Optimization Suite

Guide for evaluators

26 May, 2009

Introduction

Welcome to FICO™ Xpress! This guide provides a framework for evaluating Xpress. Using this guide you will be able to:

1. Verify that the Xpress installation was successful.

2. Decide which Xpress products to evaluate.

3. Evaluate Xpress.

1   Verify that the Xpress installation was successful

You can test that the Xpress installation was successful by launching Xpress-IVE (Windows operating system only) or by running console commands (all operating systems).

1. *Launch Xpress-IVE:* Xpress-IVE is the integrated visual development environment for Windows. To run Xpress-IVE, double click on the Xpress-IVE icon on your desktop or select *Start ≫ Programs ≫ Xpress ≫ Xpress-IVE*. You may also start up Xpress-IVE by typing `ive` in a DOS window or by double clicking onto an Xpress-Mosel model file (file with extension `.mos`).

2. *Run console commands:* At the command prompt[1], type the following sequence of commands:

    ```
    optimizer
    ```
    ⏎ (Enter)
    ```
    quit
    ```

    You will see output that looks like the following:

    ```
    C:\> optimizer
    Xpress-Optimizer v19.00.00
    Hyper capacity, MIP (20 threads), Barrier (20 threads), Network, QP/MIQP
    (c) Copyright Fair Isaac Corporation 2008

    Using Xpress-Optimizer [C:\XpressMP\bin\xprs.dll]
    Enter problem name >
    [xpress C:\] quit
    ```

---

[1]To obtain a command prompt under Windows select *Start ≫ Programs ≫ Accessories ≫ Command Prompt*

2 Decide which Xpress products to evaluate

In order to make this decision you must consider the type of problem that you wish to solve, the tools that you wish to use for model development, the tools that you wish to use for model deployment, and the platform you intend to use.

2.1 Type of problem

Xpress has solver engines for different types of problems:

Type of problem	Solving engine
Linear Programming (LP)	Xpress-Optimizer
Mixed Integer Programming (MIP)	
Quadratic Programming (QP)	
Mixed Integer Quadratic Programming (MIQP)	
Quadratically Constrained Quadratic Programming (QCQP)	
Linearly Constrained Convex Optimization (LCCO)	
Non-Linear Programming (NLP)	Xpress-SLP
Mixed Integer Non-Linear Programming (MINLP)	
Stochastic Programming (SP)	Xpress-SP
Constraint Programming (CP)	Xpress-Kalis

Further information

See Xpress documentation (in the `docs` subdirectory of your Xpress installation or at http://research-rules.fico.com/):

- Classification of mathematical programming problems: "Getting Started with Xpress", 1.1. 'Mathematical Programming'.
- Modeling with Mosel, in particular LP/MIP: "Xpress-Mosel User Guide".
- Formulation of mathematical programming problems: see the book: "Applications of Optimization with Xpress-MP", Part I: 'Developing Linear and Integer Programming models'.
- Stochastic Programming: "Overview of Stochastic Programming Applications" whitepaper.
- Constraint Programming: "Xpress-Kalis User Guide".

2.2 Tools for model development and specification

Xpress provides the following tools for model development and specification:

- **Xpress-Mosel/Xpress-IVE:** Mosel is an advanced modeling and programming language. Xpress-IVE is the integrated visual environment to develop Mosel models.
- **Xpress-BCL:** Object-oriented library callable from supported programming languages.
- **Xpress-Optimizer libraries:** Optimizer functions and procedures provided for low-level integration with applications developed in supported programming languages.

Also, a model may be specified by a matrix in MPS or LP format.

Further information

See the whitepaper "Modeling with Xpress" (available at
http://research-rules.fico.com/).

2.3 Tools for model deployment

Model deployment strongly depends on the tool used to specify the model. Deployment is achieved through the use of Xpress libraries.

Model specified in	Deployed using
Mosel	Mosel libraries (C++/C, Java, VB, .NET)
BCL	BCL libraries (C, C++, Java, VB, .NET)
Optimizer libraries	Optimizer libraries (C++/C, Java, VB, .NET)

Note that for Stochastic Programming (Xpress-SP) and for Constraint Programming (Xpress-Kalis), the model must be specified using Xpress-Mosel. Also, for Xpress-SLP deployed with Java and .NET please contact Xpress Support. For Xpress-BCL deployed with .NET please contact Xpress Support.

Further information

See Xpress documentation (in the `docs` subdirectory of your Xpress installation or at http://research-rules.fico.com/):

- Introductory examples (Mosel/BCL/Optimizer): "Getting Started with Xpress".
- Mosel libraries examples: "Xpress-Mosel User Guide", Part III: 'Working with the Mosel Libraries'.
- Complete application examples (Mosel): "Embedding optimization algorithms" whitepaper (available at
  http://research-rules.fico.com/).
- Mosel libraries documentation: "Mosel Library Reference Manual".
- BCL: "BCL Reference Manual".
- Optimizer: "Optimizer Reference Manual".

2.4 Platform

Xpress-IVE is available on Microsoft Windows platforms only. Xpress-Kalis is available on Microsoft Windows, Linux, and Solaris, all 32-bit only. All other products are available on every supported platform.

3 Evaluate Xpress

The following four typical evaluator scenarios are defined depending on the choice of products to evaluate (see question 2):

- **Scenario 1:** Develop the model in Mosel for any kind of problem and deploy in any programming language.

Tool for model development:	Mosel
Type of problem:	Any
Tool for model deployment:	Mosel Libraries (C++/C, Java, VB, .NET)

- **Scenario 2:** Develop the model in BCL for problems requiring Xpress-Optimizer and deploy in any programming language.

Tool for model development:	BCL
Type of problem:	LP, MIP, QP, MIQP
Tool for model deployment:	BCL Libraries (C, C++, Java, VB, .NET)

- **Scenario 3:** Use your custom application to develop the model and then call the Xpress-Optimizer libraries. Available for problems that can be solved with Xpress-Optimizer.

Tool for model development:	Custom application that calls optimizer libraries.
Type of problem:	LP, MIP, QP, MIQP, NLP, MNLP
Tool for model deployment:	Optimizer libraries (C++/C, Java, VB, .NET)

- **Scenario 4:** Run a matrix that is readily available in LP or MPS format. Such a model may be executed through the Xpress-IVE, console commands, or applications that call Xpress-Optimizer library functions.

Each scenario defines a sequence of specific evaluation steps.

3.1 Evaluation steps for Scenario 1

Tool for model development:	Mosel
Type of problem:	Any
Tool for model deployment:	Mosel Libraries (C++/C, Java, VB, .NET)

This scenario uses examples from the "Getting Started with Xpress" document that can be found in directory `\XpressMP\examples\getting_started\Mosel`.

3.1.1 Launch Xpress-IVE

Xpress-IVE is the integrated visual development environment for Windows. To run Xpress-IVE, double click on the Xpress-IVE icon on your desktop, select *Start ≫ Programs ≫ Xpress ≫ Xpress-IVE*. Otherwise, you may also start up IVE by typing `ive` in a DOS window or by double clicking on a model file (file with extension `.mos`).

3.1.2 Open Mosel model in Xpress-IVE

To open a Mosel model file select *File ≫ Open*. Locate the directory containing the evaluation examples (`Examples\Getting Started\Mosel`) and open the file `foliolp.mos`. The model "Portfolio optimization with LP" will open in the central pane (the IVE editor). This model seeks

an optimal investment portfolio using ten securities (shares), subject to some risk and regional constraints. The comments in the code (text following the exclamation mark '!') describe the meaning of the different statements. Note that `RET` is an array of real values representing the expected return of the shares. The decision variables in the model are given by the array `frac` of type `mpvar`. The procedure `maximize` calls Xpress-Optimizer to maximize the objective function. The code also contains statements to print the optimal solution and solution values.

Further information

- For more information on the model formulation for this example see: "Getting Started with Xpress", Chapter 2: 'Building models'.

- For more information on the Mosel representation of the model for this example see: "Getting Started with Xpress", 3.2 'LP model'.

- For more information on the Xpress-IVE editor see Xpress-IVE Help (Select *Help* ≫ *Xpress-IVE Help* ≫ *The Editor*).

- For more information on Mosel and other Mosel examples see "Xpress-Mosel User Guide", Chapter 1: 'Getting started with Mosel', 1.1 'Entering a model'.

3.1.3   Compile and run the Mosel model

To compile the Mosel model select *Build* ≫ *Compile*. The status of the compilation is shown at the bottom of Xpress-IVE (Build view of the Info Bar), and it should read *foliolp.mos compiled successfully*. Upon successful compilation, the Project Bar on the left is populated with the entity tree, which contains all the identifiers in the Mosel model. Successful compilation also creates the compiled Mosel file `foliolp.bim`

Further information

- For more on compilation and compilation errors see: "Getting Started with Xpress", 3.3. 'Correcting errors and debugging a model'.

- For more on Xpress-IVE see the Xpress-IVE Help: Project Bar, Info Bar.

To run the Mosel model select *Build* ≫ *Run*. If it was not previously compiled, running the model will also compile it. Upon successful execution, the right window (Run Bar) displays output from the Mosel code and from the Xpress-Optimizer on its various tabs. In particular, the Output/Input tab shows the output printed by the solution printing statements in the Mosel model, and the Stats and Matrix tabs show output from the Xpress-Optimizer. Also, the values of the entity tree identifiers are displayed on tooltips or view dialogs (when clicking on identifier) after the problem is solved.

String indices: The model data and solution are more easily understandable when using string indices. Open and run model `foliolps.mos`. Explore output tabs and the entity tree.

Further information

- For more information on running the Mosel model for this example see: "Getting Started with Xpress", 3.4. 'Solving, optimization displays and viewing the solution'.

- For more information on Xpress-IVE see the Xpress-IVE *Help* ≫ *Run Bar tabs/panes*.

- For more information on string indices for this example see: "Getting Started with Xpress", 3.4.1. 'String indices', and also "Xpress-Mosel User Guide", 2.1.3. 'The burglar problem revisited'.

3.1.4   Work with data in Mosel

In Mosel models you may work with a large variety of data sources, ranging from simple text files and other file formats such as databases to data exchanged in memory between a Mosel model and a host application or between several concurrent Mosel models. We show here the most frequent cases, namely text files and database access via ODBC.

Text files

Open and run model `foliodata.mos`. Note that this model has a `parameters` block and an `initializations from` block. The parameters include the data input file, output file, and other model constants. Parameters can be reset at run time, and they are particularly important when a model is deployed within a business application. The `initializations from` block reads data from the file `folio.dat`. This file has a Mosel-specific format that can be seen by opening it in Xpress-IVE: (*File ≫ Open ≫ Files of type: data file (*.dat)*). The index sets and data arrays in the model are created dynamically based on the data in the input file. Finally this model directs the solution output to an external free-format file `result.dat` by calling the procedures `fopen` and `fclose`.

Further information

For more information on working with data and using parameters in Mosel see: "Getting Started with Xpress", Chapter 4: 'Working with data'. See also the Xpress-IVE wizard for parameters, data input, and text output.

Spreadsheets and databases

Open and run model `folioexcel.mos`. The `initializations from` block in this program reads data from an MS Excel spreadsheet. The model is accompanied by data in the file `folio.xls`. Note that this example changes the sets `RISK` and `NA` into arrays of Booleans to receive the data from the file. The `initializations to` block outputs the problem results back to the spreadsheet. If the Excel file is open when writing to it the output data does not get saved, letting you choose whether to keep the results or not. Repeated model executions will overwrite previous output in the target range.

A second very similar model, `folioodbc.mos`, reads data from an ODBC data source (*e.g.*, the MS Access database `folio.mdb`) and outputs the problem results back to the database. The prefix to the filename in the `initializations` blocks now is `mmodbc.odbc`, corresponding to the type of data we work with; all else is the same as in the Excel model. The ODBC database access facility can also be employed with MS Excel spreadsheets. However, some restrictions apply and we recommend to use the Excel-specific data access as shown in the model `folioexcel.mos`. **Warning:** In order to run this example, the ODBC driver for the corresponding data source must be present. Also, in Windows you must make sure that a *User Data Source* called "MS Access Database" is setup in the *ODBC Data Source Administrator* (Windows 2000 or XP: *Start ≫ Settings ≫ Control Panel ≫ Administrative Tools ≫ Data Sources (ODBC)*).

Further information

- Using the ODBC module: "Xpress-Mosel User Guide", Chapter 2: 'Some illustrative examples' and the whitepaper "Using ODBC and other database interfaces with Mosel".

- Documentation of the ODBC module: "Mosel Language Reference Manual", Chapter 8: '*mmodbc*'.

- Overview of other possibilities of data exchange with external sources: "Xpress-Mosel User Guide", 16.1 'Generalized file handling'.

- Examples of advanced communication methods with external data sources are given in the whitepaper "Generalized file handling in Mosel".

3.1.5   Mosel MIP and quadratic models

Open and run the following Mosel models:

`foliomip1.mos`: This model introduces the array of binary variables **buy** to impose a constraint limiting the number of different shares taken into the portfolio.

`foliomip2.mos`: This model redefines the array of variables **frac** to be semi-continuous, so that at least a certain minimum amount of the budget is spent on each share that is bought.

`folioqp.mos`: This model uses a quadratic formulation to minimize the portfolio variance subject to achieving a target expected return. The Mosel program solves the problem twice, where the second run imposes a limit on the number of shares taken into the portfolio.

For each model, explore the solution and information displayed in the Run Bar tabs.

Further information

- "Getting Started with Xpress", Chapter 6: 'Mixed Integer Programming' and Chapter 7: 'Quadratic Programming'.

- Complete list of available MIP variable types: "Xpress-Mosel User Guide", Chapter 4: 'Integer Programming'.

3.1.6   Other problem types: Constraint Programming, nonlinear and stochastic models

All models we have seen so far use Xpress-Optimizer for problem solving (chosen with the statement `uses "mmxprs"` at the begin of the model). If we wish to use a different solver, we need to indicate the name of the corresponding solver module.

Open the model `assign.mos`: this model implements and solves an assignment problem with Xpress-Kalis, that is, using Constraint Programming (CP) techniques. For given sets of workers and machines the problem is to assign exactly one worker to every machine, maximizing the total productivity. The productivity of a worker depends on the machine he is assigned to.

You may observe several differences to the models we have seen previously:
– The solver choice statement now is `uses "kalis"`.
– The CP decision variables are of the type `cpvar`; their domain (=admissible values) can be set with the procedure `setdomain`.
– CP models may have linear constraints (as in the 'Total productivity' constraint), however our model also uses other types of constraint relations, so-called 'global constraints'. The `element`

constraint formulates a discrete function in one variable, and the `all_different` relation states that all variables in the constraint need to take a different value.
– The CP problem is solved with tree search methods. Instead of using the default search strategies, it is usually preferable to choose a more problem-specific strategy (using procedure `cp_set_branching`).
– The function `cp_maximize` is used to invoke the optimization.
All else (general structure, declarations, access to data, output printing) remains unchanged from what we have seen so far.

When running this model with IVE the entity tree display is populated and the model output appears in the *Output/Input* pane at the right hand side of the workspace. Detailed information about the model execution can be found by selecting the tabs *CP stats* (summary problem statistics) and *CP search* (graphical representation of the CP search tree).

Other solver types available for Mosel include Xpress-SLP for solving Nonlinear Programming problems and Xpress-SP for stochastic optimization problems. Each of these solver modules comes with problem-type specific functionality (such as variable and constraint types)—please see the corresponding manuals. In addition certain IVE displays are adapted to the solver(s) used by a Mosel model.

Further information

- **Xpress-Kalis** provides access to the functionality of the **Constraint Programming** solver Kalis from within the Mosel environment. For more information about Xpress-Kalis see the documents "Xpress-Kalis User Guide" and "Xpress-Kalis Reference Manual".

- **Xpress-SP** is an extension of the Mosel language for formulating **Stochastic Programming** problems. For more information about Xpress-SP see the document "Xpress Stochastic Programming Guide".

- **Xpress-SLP** is a solver for **Non-Linear** and **Mixed Integer Non-Linear Programming** based on the technique of successive linear approximations. For more information about Xpress-SLP see the document "Xpress-Mosel SLP Reference Manual".

3.1.7 Deploying Mosel models

Open a Mosel model and select *Deploy ≫ Deploy* or click the deploy button. This will open the Deploy dialog box. Select the programming language you wish to run the Mosel model from, and click the *Next>* button. This will open a Source Code Dialog containing the code for deployment.

Further information

- Xpress-IVE Help (Select *Help ≫ Xpress-IVE Help ≫ Dialogs ≫ Deploy Dialog*).

- Introductory example (VB): "Getting Started with Xpress", Chapter 10: 'Embedding a Mosel model in an application'.

- Further examples (C/Java/VB): "Xpress-Mosel User Guide", Part III: 'Working with the Mosel Libraries'.

- Documentation of Mosel C libraries: "Xpress-Mosel Library Reference Manual".

- Documentation of Mosel Java libraries: "Xpress-Mosel Library Reference Manual JavaDoc".

- Documentation of the Mosel .NET interface: "Xpress-Mosel .NET Interface".

3.1.8   Exporting matrix files

After executing a Mosel model select Build ≫ Export matrix file in order to generate a file with an LP or MPS representation of the model.

Further information

- Xpress-IVE Help (select *Help ≫ Xpress-IVE Help ≫ Dialogs ≫ Export to Matrix Dialog*).
- "Getting Started with Xpress", 9.4 'Matrix files'.
- Xpress-IVE Help (select *Help ≫ Xpress-IVE Help ≫ Dialogs ≫ Optimizer Dialog*).

3.1.9   Mosel console commands

As an alternative to running Mosel models within Xpress-IVE you may execute them with the Mosel standalone version. This mode is often preferrable for testing and experimentation, for instance, if you wish to invoke a sequence of model runs from a batch file.

At the command prompt, type the following sequence of commands:

```
mosel
compile foliolp
load foliolp
run
quit
```

You will see output that looks like the following:

```
mosel
 ** Xpress-Mosel **
 (c) Copyright Fair Isaac Corporation 2008
>compile foliolp
 Compiling 'foliolp'...
>load foliolp
>run
 Total return: 14.0667
 treasury: 30%
 hardware: 0%
 theater: 20%
 telecom: 0%
 brewery: 6.66667%
 highways: 30%
 cars: 0%
 bank: 0%
 software: 13.3333%
 electronics: 0%
 Returned value: 0
>quit
 Exiting.
```

The command sequence above may be shortened to a single line:

```
mosel -c "exec foliolp"
```

Further information

See "Xpress-Mosel User Guide", 1.1 'Entering a model'. See also "Xpress-Mosel Reference

Manual", 1.1 'What is Mosel' – 'Running Mosel'.

3.1.10   Other Mosel topics

In addition to the topics addressed in this guide, the document "Getting Started with Xpress" describes how to draw **user graphs** in Xpress-IVE (Chapter 5: 'Drawing user graphs') and how to **program heuristics** with Mosel (Chapter 8: 'Heuristics'). Further details on these topics can be found in the "Xpress-Mosel User Guide", Part II 'Advanced language features' that describes Mosel's programming facilities and includes other examples of heuristics, and under Part IV 'Extensions and tools' – 16.3 'Graphics with mmive and mmxad'.

Xpress Application Developer (XAD) extends the functionality of Mosel with a set of functions and procedures for creating standard graphical user interfaces. To learn more about XAD see the document "Xpress Application Development Reference Manual".

Additional examples of modeling and programming with Mosel can be found in the directory `\XpressMP\examples\`. Also, the book "Applications of Optimization with Xpress-MP" (Dash Optimization, 2002) shows how to formulate and solve a large number of application problems with Xpress: http://research-rules.fico.com/

3.2   Evaluation steps for Scenario 2

Develop a model in BCL for problems requiring Xpress-Optimizer and deploy in any programming language.

Tool for model development:	BCL
Type of problem:	LP, MIP, QP, MIQP
Tool for model deployment:	BCL Libraries (C, C++, Java, VB, .NET)

See "Getting Started with Xpress", Part II: 'Getting started with BCL', which presents examples in C++ language with detailed explanation. The examples are also implemented in C, Java, and VB and they can be found in the corresponding directories `\XpressMP\examples\bcl\*\UGExpl`.

Further information

More examples and the complete documentation of BCL can be found in the "Xpress-BCL Reference Manual" and the "Xpress-BCL Javadoc".

3.3   Evaluation steps for Scenario 3

Use your custom application to develop the model and then call the Xpress-Optimizer libraries. Available for problems that can be solved with Xpress-Optimizer

Tool for model development:	Custom application that calls optimizer libraries.
Type of problem:	LP, MIP, QP, MIQP, NLP, MNLP
Tool for model deployment:	Optimizer libraries (C++/C, Java, VB, .NET)

See "Getting Started with Xpress", Part III: 'Getting started with the Optimizer', which presents examples in C language with detailed explanation.

Further information

- "Xpress-Optimizer Reference Manual"
- For Non-linear programming (NLP,MNLP) see "Xpress-SLP Program Reference Manual".

3.4 Evaluation steps for Scenario 4

Run a matrix that is readily available in LP or MPS format. Such a model may be executed through Xpress-IVE, console commands, or applications that call Xpress-Optimizer library functions.

3.4.1 Xpress-IVE

Select *File ≫ Open* to open file with matrix file (usually files with extensions `.mat`, `.mps`, and `.lp`). Select *Build ≫ Optimize Matrix File* to execute problem.

Further information

- Xpress-IVE Help (Select *Help ≫ Xpress-IVE Help ≫ Dialogs ≫ Optimizer Dialog*).
- "Getting Started with Xpress", 9.4. 'Matrix files'

3.4.2 Console commands

At the command prompt, type the following sequence of commands to execute the MPS matrix in file foliolp.mps:

```
optimizer
foliolp
readprob
maxim
printsol
quit
```

You will see output that looks like the following:

```
>optimizer
 Xpress-Optimizer v19.00.00
 Hyper capacity, MIP (20 threads), Barrier (20 threads), Network, QP/MIQP
 (c) Copyright Fair Isaac Corporation 2008
 Using Xpress-Optimizer [C:\XpressMP\bin\xprs.dll]
 Enter problem name > foliolp
 [xpress C:\] readprob

 Reading Problem moselP

 Problem Statistics
          4 (      0 spare) rows
         10 (      0 spare) structural columns
         29 (      0 spare) non-zero elements
 Global Statistics
          0 entities      0 sets       0 set members
 [xpress C:\] maxim
 Presolved problem has:     3 rows      10 cols      19 non-zeros

   Its         Obj Value    S   Ninf  Nneg       Sum Inf  Time
     0         42.600000    D      2     0       3.166667     0
```

```
        5         14.066659      D      0      0        .000000      0
Uncrunching matrix
        5         14.066659      D      0      0        .000000      0
Optimal solution found
[xpress C:\] printsol

Problem Statistics
Matrix moselP
Objective *OBJ*

RHS *RHS*
Problem has      4 rows and      10 structural columns

Solution Statistics
Maximization performed
Optimal solution found after      5 iterations
Objective function value is    14.066659
type c/r to continue, anything else to finish >

Rows Section
    Number   Row      At      Value      Slack Value   Dual Value      RHS
  N   1   *OBJ*      BS     14.066659    -14.066659     .000000      .000000
  E   2   _R1        EQ      1.000000      .000000      8.000000     1.000000
  G   3   _R2        LL       .500000      .000000     -5.000000      .500000
  L   4   _R3        UL       .333333      .000000     23.000000      .333333
type c/r to continue, anything else to finish >

Columns Section
    Number   Column   At      Value      Input Cost   Reduced Cost
  C   5   frac(1)    UL       .300000     5.000000      2.000000
  C   6   frac(2)    LL       .000000    17.000000     -9.000000
  C   7   frac(3)    BS       .200000    26.000000       .000000
  C   8   frac(4)    LL       .000000    12.000000    -14.000000
  C   9   frac(5)    BS       .066667     8.000000       .000000
  C  10   frac(6)    UL       .300000     9.000000      1.000000
  C  11   frac(7)    LL       .000000     7.000000     -1.000000
  C  12   frac(8)    LL       .000000     6.000000     -2.000000
  C  13   frac(9)    BS       .133333    31.000000       .000000
  C  14   frac(10)   LL       .000000    21.000000    -10.000000
[xpress C:\] quit
```

Further information

Xpress-Optimizer Reference Manual", Chapter 6: 'Console and Library functions'.

3.4.3   Xpress-Optimizer library functions

See "Getting Started with Xpress", Chapter 14: 'Matrix input', which presents an example in C language with detailed explanation.