Xpress の基礎

本書の全ての問題は、Xpress-Mosel（短縮して Mosel）言語で記述されています。問題を解く場合ユーザは、Mosel コマンド文か Xpress-IVE を選択できます。本書の最適化問題を解く場合、線形問題や混合整数計画問題を Mosel 言語で記述したモデルを解きます。本書の中で述べるレファレンス・マニュアルや例題の完成版は下記のアドレスからダウンロード可能です。
http://www.msi-jp.com/xpress/learning/square/

次に Mosel 言語の基礎を紹介致します。

1.例題の紹介

ある差し物師が、‘つげ材’を使って 2 種類の箱を作ります。小箱には、‘こまい’に 3 時間かかり、大箱には、2 時間の‘こまい’がかかります。差し物技術を持った 4 名が、週 40 時間の‘こまい’作業にかかり、合計 160 時間の作業時間がかかります。小箱は‘つげ材’が 1kg 必要であり、大箱は、3kg の‘つげ材’が必要です。‘つげ材’は、貴重品で週当たり 200kg しか得られません。大箱の売値は、20＄で、小箱は、5＄です。毎週どのサイズの箱をそれぞれ作れば売り上げが最大になるかを決めて下さい。

ここからは、“chess.mos”モデルの説明に入ります。（Mosel では、モデル名の接尾語に標準として“mos”を使います）

```
model Chess
uses "mmxprs" ! We shall use Xpress-Optimizer
declarations
xs, xl: mpvar ! Decision variables: produced quantities
end-declarations
Profit:= 5*xs + 20*xl ! Objective function
Boxwood:= 1*xs + 3*xl <= 200 ! kg of boxwood
Lathe:= 3*xs + 2*xl <= 160 ! Lathehours
maximize(Profit) ! Solve the problem
writeln("LP Solution:") ! Solution printing
writeln(" Objective: ", getobjval)
```

```
writeln("Make ", getsol(xs), " small sets")
writeln("Make ", getsol(xl), " large sets")
end-model
```

Xpress-Mosel の使い方

Mosel は、モデル開発と解を出す先取的な言語と環境を備えております。また、Optimizer は、問題を明確に捕らえ究極の精度で解を出します。Mosel は、最適化問題のモデルを開発出来る強力な簡易言語です。Mosel は、違ったフォーマットのデータをモジュール体系で読み込むことが出来ます。（Excel データやテキスト形式のデータを読み込みます。また、解を各種のフォーマット形式に出力します。）解は、厳密解か局所解です。Mosel は、大規模なアプリケーションに組み込んだモデルをプログラミング言語でライブラリーを呼び出し解くことが可能な機能を有しております。“chess.mos”モデルをコマンド文解法で解く方法は、コマンド・プロンプト画面で次のコマンドを入力して下さい。

```
mosel
exec chess
quit
```

もしモデルに文法エラーが無い場合、Mosel が起動して、コンパイルをし、解を得て停止します。アウトプットは、Mosel アウトプット形式で下記のように太字で出ます。

```
** Xpress-Mosel **
(c) Copyright Dash Associates 1998-2006
> exec chess
LP Solution:
Objective: 1333.33
Make 0 small sets
Make 66.6667 large sets
Returned value: 0
> quit
Exiting.
```

同じ解法を次のコマンドで一挙に処理できます。

```
mosel -c "exec chess"
```

“-c”は、“　”内のコマンド文に従属することを示しております。

Mosel が処理を開始して解釈出来ないコマンド文（例；h）を入力しますと Mosel は、簡単な説明文を付けてコマンド文全リストを表示致します。（もしくは、有効なコマンド文を表示します）別のオプションとしてＯＳのコマンド文から

“mosel –h”

と入力してもコマンド文全リストを表示致します。

Mosel は、言語に特別な function を追加出来ように幾つもの modules を持っております。ある module の全 function リストを“exam”コマンド文で出すことが出来ます。

例えば、Xpress-Optimizer の”module mmxprs“の内容は次の様に入力してください。

```
mosel -c "exam mmxprs"
```

Mosel 言語と Mosel コマンド文のリンク処理を完全に記述すために下記のアドレスから Reference Manual を参照して下さい。

http://www.msi-jp.com/xpress/learning/square/

上記のアドレスから Mosel の個々のマニュアルをダウンロードできます。

IVE の使い方

Xpress-IVE は、単に IVE とも言いますが、ビジュアル開発環境（Xpress **I**nteractive **V**isual **E**nvironment) のことで、Microsoft Windows 下で完全なモデル開発や最適化計算を行います。

Mosel は、組み込まれたテキスト編集機能をもとに容易に使える G raphical User Interface (GUI) 機能を備えています。IVE は、複数のモデルを開発・管理し、解くことも出来ます。また、プロトタイプモデル開発・デバッグ出来る理想的な開発環境を提供します。“chess.mos”モデル・ファイルを解くためには、次の手順で進めて下さい。

・IVE の起動

・モデル・ファイルを *File* > *Open* で開く。ソースモデルは、Windows の中央画面に出ます。

モデル編集画面で“**IVE Editor**”と呼びます。

・ランボタン（緑の三角ボタン）押す。または、*Build* > *Run.*を押す。図 5.1 に表示されて通り

Windows の右画面に計算結果が表示されます。

Windows の下画面（**Build** pane）にコンパイルの診断結果を表示します。もし、モデルに文法エラーがある場合、モデル内の発生した文字を表示し、エラー内容を記述します。エラー部分をクリッ

クしますと Edit 画面にあるエラーラインは、ここでは黄色の幅線が引かれます。

モデルを解いた後、Windows の右画面（**Output/Input** pane）にモデル内で記述したアウトプットの内容を表示します。モデル内の Write 文で記述し各変数値の値を画面表示します。IVE は、どのような解が得られたかをグラフで表現出来ます。この手法は、最適解が出たらデフォルト形式で表示します。右画面は、解く問題の形式や特別なアルゴリズムに適した問題の解の内容を表現する窓を幾つも保持しております。IVE では、モデル内にサブルーテインを組み込んでグラフを画くことも出来ます。（詳細は、マニュアルを参照して下さい）

IVE は、Windows の左画面（**Entities** pane）に解の全ての情報を保持しております。この画面内の決定変数を開いて、マウスで 1 変数ごとクリックして見て下さい。各変数の値や Reduced Cost が表示されます。式変数には、Dual 値や Slack 値が表示されます。

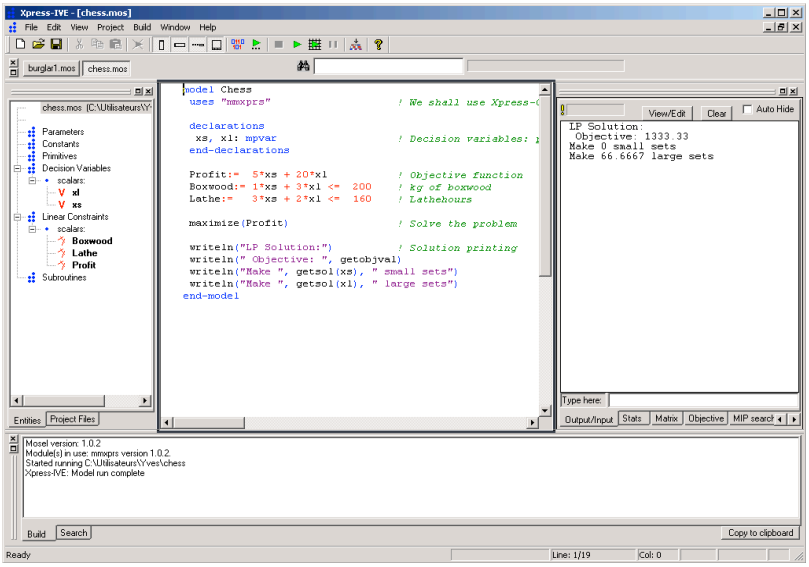

Figure 5.1: Xpress-IVE display after running model `chess.mos`

2.Mosel でモデル開発

2 つ目として、少し大きいモデル「泥棒問題」を説明します。このモデルを使って以下の章に掲載されている諸例題にも繰り返し使用する Mosel 言語の基本的な機能を説明致します。

泥棒問題

泥棒は、8 個の価値と重さの違う品物を見ている。彼は、持てるだけ重さの品物を持ち出したいがその価値が最大(WTMAX）であることを望んでいる。品物（ITEMS(i)）の中からどれを取るか

（take(i)）モデル化をします。品物(i）を取る場合 take(i)は、1 で、取らない場合 take(i)は、0 とします。take(i)は、binary の決定変数と定義します。各品物の i 番目の価値や重量をそれぞれ”VALUE(i)”、“WEIGHT(i)”と設定します。問題の式表現は、以下の通りです。

maximize Σ VALUEi・takei (maximize the total value)
i ∈ ITEMS
Σ WEIGHTi・takei ≦ WTMAX (weight restriction)
i ∈ ITEMS
∀*i* ∈ *ITEMS* : *takei* 2 {0, 1}

この問題は、ナップサックの例題です。Mosel では、次のように記述します。

```
model "Burglar 1"
uses "mmxprs"
declarations
ITEMS = 1..8 ! Index range for items
WTMAX = 102 ! Maximum weight allowed
VALUE: array(ITEMS) of real ! Value of items
WEIGHT: array(ITEMS) of real ! Weight of items
take: array(ITEMS) of mpvar ! 1 if we take item i; 0 otherwise
end-declarations
! Item:     1    2    3    4   5    6    7  8
VALUE := [15, 100, 90, 60, 40, 15, 10, 1]
WEIGHT:= [ 2, 20, 20, 30, 40, 30, 60, 10]
! Objective: maximize total value
MaxVal:= sum(i in ITEMS) VALUE(i)*take(i)
! Weight restriction
sum(i in ITEMS) WEIGHT(i)*take(i) <= WTMAX
! All variables are 0/1
forall(i in ITEMS) take(i) is_binary
maximize(MaxVal) ! Solve the MIP-problem
! Print out the solution
writeln("Solution:\n Objective: ", getobjval)
forall(i in ITEMS) writeln(" take(", i, "): ", getsol(take(i)))
end-model
```

モデルの解は次の通りです。

```
Solution:
Objective: 280
```

```
take(1): 1
take(2): 1
take(3): 1
take(4): 1
take(5): 0
take(6): 1
take(7): 0
take(8): 0
```

このモデルや一般的な Mosel モデルの構成は次の通りです。

・**Model:** Mosel のプログラムは、全て Keyword “model”で始まります。その後にモデル名を書きます。モデルの終了は、Keyword “end-model”で閉じます。

・**Declarations:** 全ての object は、declarations block で定義しなければなりません。但し、assignment 方法で値を明白に定義すること出来ます。（例：i::1 とは、“i”が整数の“1”の値を取ることを設定しています。例題の目的関数“MaxVal”は、線形値を取ると設定されています。

・**Problem definition:** 特にモデルは、データの仕様で起動します。（ここでは、”VALUE”と“WEIGHT”の値を assignment します）その後に引き続き文面を記述します。ここでは、目的関数“MaxVal”の定義文・総重量の上限制約式の定義・取れる品物変数が binary(“0” or “1”)変数であることの定義と記述します。

・**Solving:** “maximize”プロシデアーを使って、Xpress-Optimizer を呼び出し、目的関数“MaxVal”を最大化します。Mosel では、”default solver”が無いのでプログラム開始前に”uses “mmxprs””の記述文を書き Optimizer が起動することを明記しております。

・**Output printing:** 最後の 2 行の文は、最適解の値と決定経変数の解の値をプリントする文です。

・**Line breaks:** 1 文内に違った制約式を記述する場合は、“；”で分離出来ます。(例： `x1 <= 4;`

`x2 >= 7`). 逆に、文面の終了や続きを表現する方法はありません。文面が次の行に続けたい場合は、

オペレータ (+, >= *etc.*) や文字“`,`”を記しますと文面が続くことを表します。

・**Comments:** 例で示しているようにシンボル！は、コメントの始まりを示し行の終わりまでをコメントとして扱います。複数行の文面を書く場合、シンボル！でコメントの始まりを示し最終行末尾にシンボル！を記します。（！　コメント　！）モデルの詳細な使い方を説明致します。

・**Ranges and sets:**

```
ITEMS = 1..8
```

レンジセットの定義の方法は、上記のように連続する 1 から 8 迄の幅を示します。このレンジ手法は、`data array(VALUE, WEIGHT)`や決定変数（`take`）に使用しています。`ITEMS = 1..8` の数値 `Index` で記述する代わりに次の様な記述法もあります。

```
ITEMS = {"camera", "necklace", "vase", "picture", "tv", "video",
"chest", "brick"} ! Index set for items
```

・**Arrays**:：

```
VALUE: array(ITEMS) of real
```

1 次元の array は、上記例のように ITEMS のレンジセットの Index で実数定義をします。

多次元の array 表記は。下記の通りです。

```
VAL3: array(ITEMS, 1..20, ITEMS) of real
```

上記の例は、3 次元の array 表示です。決定変数 take の array のタイプは、例文に記した通り、

“mpvar”です。Mosel では、全ての Object(スカラーや array)は,Default 値として次の様に定義します。

`real, integer`: 0
`boolean`: `false`
`string`: `''` (*i.e.* the empty string)

データ array の値を例題で示したようにモデル内での設定方法や 5.2.2 節で示す外部データから呼び込む方法（Initialization from)があります.

・**Summations:**

```
MaxVal:= sum(i in Items) VALUE(i)*x(i)
```

価値の総和を示す“MazVal”は、線形表示で”sum”関数で定義します。

$$\sum_{i \in ITEMS} VALUEi \cdot Xi$$

- **Simple looping:**

```
forall(i in ITEMS) take(i) is_binary
```

レンジインデックス（ITEMS）数だけ決定変数（take）を loop させます。つまり ITEMS のレンジ（1..8）を呼び出して、take(1),take(2), take(8)が binary であること記述しています。例題のように全ての解の値をプリントアウトしたい場合は、最終行の前で“forall”の手法が使えます。他の loop 形式は、Application 例題で使われています。

- **Integer Programming variable types:**

決定変数（mpvar）を binary 変数と定義するには、例えば、決定変数”xbinvar”を binary 変数と定義する方法は、次のような文面になります。

```
xbinvar is_binary
```

例えば、MIP 問題で決定変数”xbinvar”を整数変数と定義する方法は、次のような文面になります。

```
xintvar is_integer
```

Text ファイルからデータの呼び込み

次の例題は、外部のテキスト・ファイルからモデル内のテーブルにどの様に呼び込むかを示します。泥棒問題では、ITEM データをモデル内で記述しましたが、この例題は、ITEM データを外部データファイルに所持しています。外部テキスト・ファイルから呼び込む Mosel モデルの例題“burglar2.mos”を次の通り示します。

```
model "Burglar 2"
uses "mmxprs"
declarations
ITEMS: set of string ! Set of items
WTMAX = 102 ! Maximum weight allowed
```

```
VALUE: array(ITEMS) of real ! Value of items
WEIGHT: array(ITEMS) of real ! Weight of items
end-declarations
initializations from 'burglar.dat'
VALUE
WEIGHT
end-initializations
declarations
take: array(ITEMS) of mpvar ! 1 if we take item i; 0 otherwise
end-declarations
! Objective: maximize total value
MaxVal:= sum(i in ITEMS) VALUE(i)*take(i)
! Weight restriction
sum(i in ITEMS) WEIGHT(i)*take(i) <= WTMAX
! All variables are 0/1
forall(i in ITEMS) take(i) is_binary
maximize(MaxVal) ! Solve the MIP-problem
! Print out the solution
writeln("Solution:\n Objective: ", getobjval)
forall(i in ITEMS) writeln(" take(", i, "): ", getsol(take(i)))
end-model
```

外部のテキスト・ファイル“ burglar.dat”内容は次の通りです。

```
VALUE: [("camera") 15 ("necklace") 100 ("vase") 90 ("picture") 60
("tv") 40 ("video") 15 ("chest") 10 ("brick") 1]
WEIGHT:[("camera") 2 ("necklace") 20 ("vase") 20 ("picture") 30
("tv") 40 ("video") 30 ("chest") 60 ("brick") 10]
```

Mosel では、“initialization block”でどこから外部データを呼び込むかを設定します。ファイル内のデータ項目の順番は、“initialization block”内の項目と同じでなくても問題ありません。ITEMS の内容は、“VALUE”及び“WEIGHT”の Index を通じて間接的に数値を呼び込みます。外部データを呼び込む“initialization”コマンドを使えば ITEMS の数値は、既知となります。

Application 例題にあるように、データや決定変数を“dynamic array”でどのように表現するかを示しています。

データを“KNAPSACK”ファイルから、次のように外部データファイルから呼び込めます。

```
initializations from 'burglar2.dat'
[VALUE, WEIGHT] as 'KNAPSACK'
end-initializations
```

“burglar2.dat”のデータファイルの内容は、 次の通りです。

```
KNAPSACK:
[ ("camera") [ 15 2]
("necklace") [100 20]
 ("vase") [ 90 20]
("picture") [ 60 30]
("tv") [ 40 40]
("video") [ 15 30]
("chest") [ 10 60]
("brick") [ 1 10] ]
```

本書の例題は、常に外部データをテキスト・ファイルから呼び込みます。しかし、Mosel では、他のソフトからのデータを呼び込むことや解の値を書き出すことが出来ます。Excel や Access のソフトまた、メモリ内のデータも呼び込むことが出来ます。詳細はマニュアルを参照して下さい。

予約語

Mosel の予約語は、次の通りです。上記例題記述のバージョンも予約語は同じです。

（例：”AND”及び”and”は、kyeword です。“And”は、kyeword ではありません）組み込み用として使用する以外は、モデル開発で書き予約語を使わないで下さい。

and, array, as
boolean, break
case
declarations, div, do, dynamic
elif, else, end
false, forall, forward, from, function
if, in, include, initialisations, initializations, integer, inter,
is_binary, is_continuous, is_free, is_integer, is_partint, is_semcont,
is_semint, is_sos1, is_sos2
linctr
max, min, mod, model, mpvar
next, not
of, options, or
parameters, procedure, public, prod
range, real, repeat
set, string, sum
then, to, true、union, until, uses、while