

# Rapid LP Development with Mosel

Tom Halliburton  
Energy Modeling Consultants Ltd.

for

Operations Research Society of New Zealand  
November, 2003

# Building Linear Programs

- Interface options
- Dash Optimisation
- Dash product range
- Main features of Mosel
- Example problem

# High Level Math Programming Interfaces

- Modeling Languages
  - GAMS
  - AMPL
  - LINGO
  - MPL
- Excel Solver
- Programming APIs
  - ILOG Planner & Concert
  - Xpress BCL
- Recent Modeling Languages
  - ILOG OPL
  - Xpress Mosel

# Dash Optimisation

- Owned by Bob Daniels & Robert Ashford
- Offices in UK and US
- Established in 1980s
- Solvers available:
  - LP primal, dual, interior point
  - MIP
  - Quadratic Programming and MIP + QP
  - Successive approximations LP, with MIP
- Programming API for C, VB, Java - BCL
- Modeling language - Mosel



# Mosel Overview

- Developed by Dash, in France
- IVE development environment written in New Jersey
- First released 2001
- Interfaces directly with all Dash solvers
  - not via files
- More like a programming language than AMPL
- Graphical development environment for Windows
  - edit, compile, run, debug within the IDE
- Command line interface for other operating systems
- Provides a compiled model file for distribution
  - preserves intellectual property
  - facilitates run time licensing
- Can be embedded in C, VB, Java code

# Why Mosel?

- Xpress solvers amongst the best
- Price competitive
  - Mosel, IVE, LP, MIP: US\$9900
- Flexibility of licensing
- Good telephone & email support
- Successful development using Mosel v1.0.1
  - early versions of software sometimes risky
  - no gotchas
- Licensing procedures usually straight forward
- Documentation good
- Stick with something I trust
  - time required to learn new systems, languages, etc

# Programming Constructs

- forall loop
- while
- if then else
- case
- repeat until
- functions & subroutines
- sets & set operations
- read from text, spreadsheets or databases
- dynamic and sparse arrays, with “exists” function

# The Burglar Problem

declarations

Items={"camera", "necklace", "vase", "picture", "tv", "video",  
"chest", "brick"} ! Index set for items  
VALUE: array(Items) of real ! Value of items  
WEIGHT: array(Items) of real ! Weight of items  
WTMAX=102 ! Max weight allowed  
x: array(Items) of mpvar ! 1 if we take item i; 0 otherwise

end-declarations

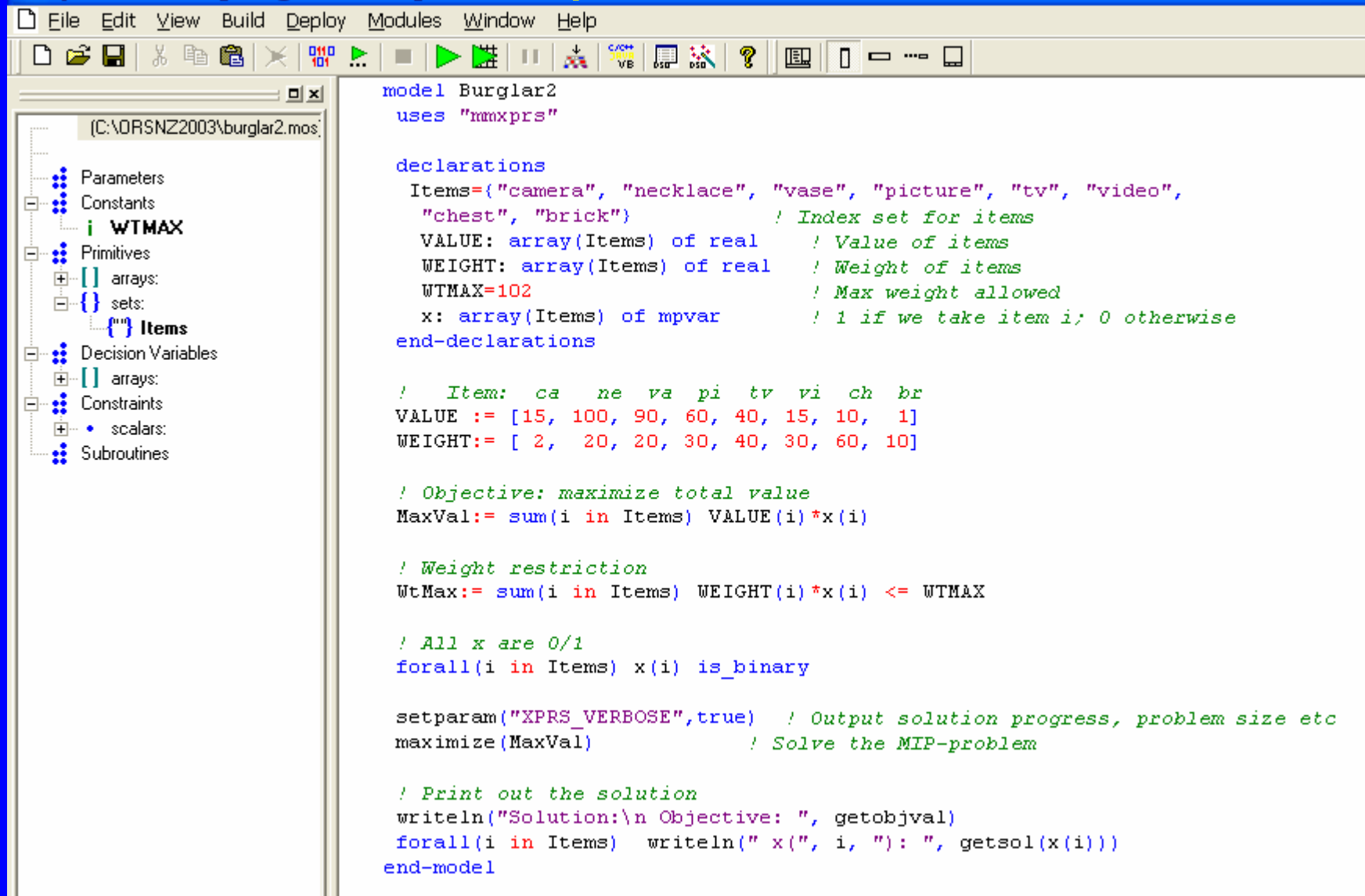
VALUE := [15, 100, 90, 60, 40, 15, 10, 1]  
WEIGHT:= [ 2, 20, 20, 30, 40, 30, 60, 10]

MaxVal:= sum(i in Items) VALUE(i)\*x(i)  
WtMax:= sum(i in Items) WEIGHT(i)\*x(i) <= WTMAX  
forall(i in Items) x(i) is\_binary

maximize(MaxVal)



# IVE - Development Environment



The screenshot displays the IVE Development Environment interface. On the left, a project tree shows the structure of the model file 'burglar2.mos'. The tree includes sections for Parameters, Constants (with 'WTMAX' highlighted), Primitives (containing arrays and sets, with 'Items' highlighted), Decision Variables, Constraints, and Subroutines. The main editor window on the right contains the following GAMS code:

```
model Burglar2
uses "nmxprs"

declarations
  Items={"camera", "necklace", "vase", "picture", "tv", "video",
    "chest", "brick"}          ! Index set for items
  VALUE: array(Items) of real   ! Value of items
  WEIGHT: array(Items) of real  ! Weight of items
  WTMAX=102                     ! Max weight allowed
  x: array(Items) of mpvar      ! 1 if we take item i; 0 otherwise
end-declarations

! Item:  ca  ne  va  pi  tv  vi  ch  br
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)*x(i)

! Weight restriction
WtMax:= sum(i in Items) WEIGHT(i)*x(i) <= WTMAX

! All x are 0/1
forall(i in Items) x(i) is_binary

setparam("XPRS_VERBOSE",true) ! Output solution progress, problem size etc
maximize(MaxVal)               ! Solve the MIP-problem

! Print out the solution
writeln("Solution:\n Objective: ", getobjval)
forall(i in Items)  writeln(" x(", i, "): ", getsol(x(i)))
end-model
```

# Viewing Data - move mouse over name

The screenshot shows the GAMS (General Algebraic Modeling System) software interface. The left-hand pane displays a project tree for a file named 'burglar2.mos'. The tree is organized into categories: Parameters, Constants, Primitives, arrays, sets, and Items. The 'Items' set is currently selected, and a tooltip is displayed over it, indicating it contains 8 elements of type string: 'camera', 'necklace', 'vase', 'picture', 'tv', 'video', 'chest', and 'brick'. The main window shows the GAMS code for a model named 'Burglar2'. The code includes a 'uses' statement for 'nmxprs', a 'declarations' section defining the 'Items' set and various arrays, and a 'model' section defining the objective function and constraints. The code is as follows:

```
model Burglar2
  uses "nmxprs"

  declarations
    Items={"camera", "necklace", "vase", "picture", "tv", "video",
           "chest", "brick"}          ! Index set for items
    VALUE: array(Items) of real       ! Value of items
    WEIGHT: array(Items) of real      ! Weight of items
    WTMAX=102                         ! Max weight allowed
    x: array(Items) of mpvar          ! 1 if we take item i; 0 otherwise
  end-declarations

  Item: ca ne va pi tv vi ch br
  UE := [15, 100, 90, 60, 40, 15, 10, 1]
  GHT:= [ 2,  20, 20, 30, 40, 30, 60, 10]

  bjective: maximize total value
  Val:= sum(i in Items) VALUE(i)*x(i)

  ! Weight restriction
  WtMax:= sum(i in Items) WEIGHT(i)*x(i) <= WTMAX

  ! All x are 0/1
  forall(i in Items) x(i) is_binary

  setparam("XPRS_VERBOSE",true) ! Output solution progress, problem size etc
  maximize(MaxVal)              ! Solve the MIP-problem

  ! Print out the solution
  writeln("Solution:\n Objective: ", getobjval)
  forall(i in Items) writeln(" x(", i, "): ", getsol(x(i)))
end-model
```

# Viewing Data

The screenshot shows the GAMS (General Algebraic Modeling System) software interface. The main window displays a model named 'Burglar2' with the following code:

```
model Burglar2
uses "maxprs"

declarations
  Items={"camera", "necklace", "vase", "picture", "tv", "video",
        "chest", "brick"}          / Index set for items
  VALUE: array(Items) of real      / Value of items
  HEIGHT: array(Items) of real    / Height of items

end-model
```

A 'View/Edit text' dialog box is open, showing the contents of the 'Items' set:

Items contains 8 elements of type string:

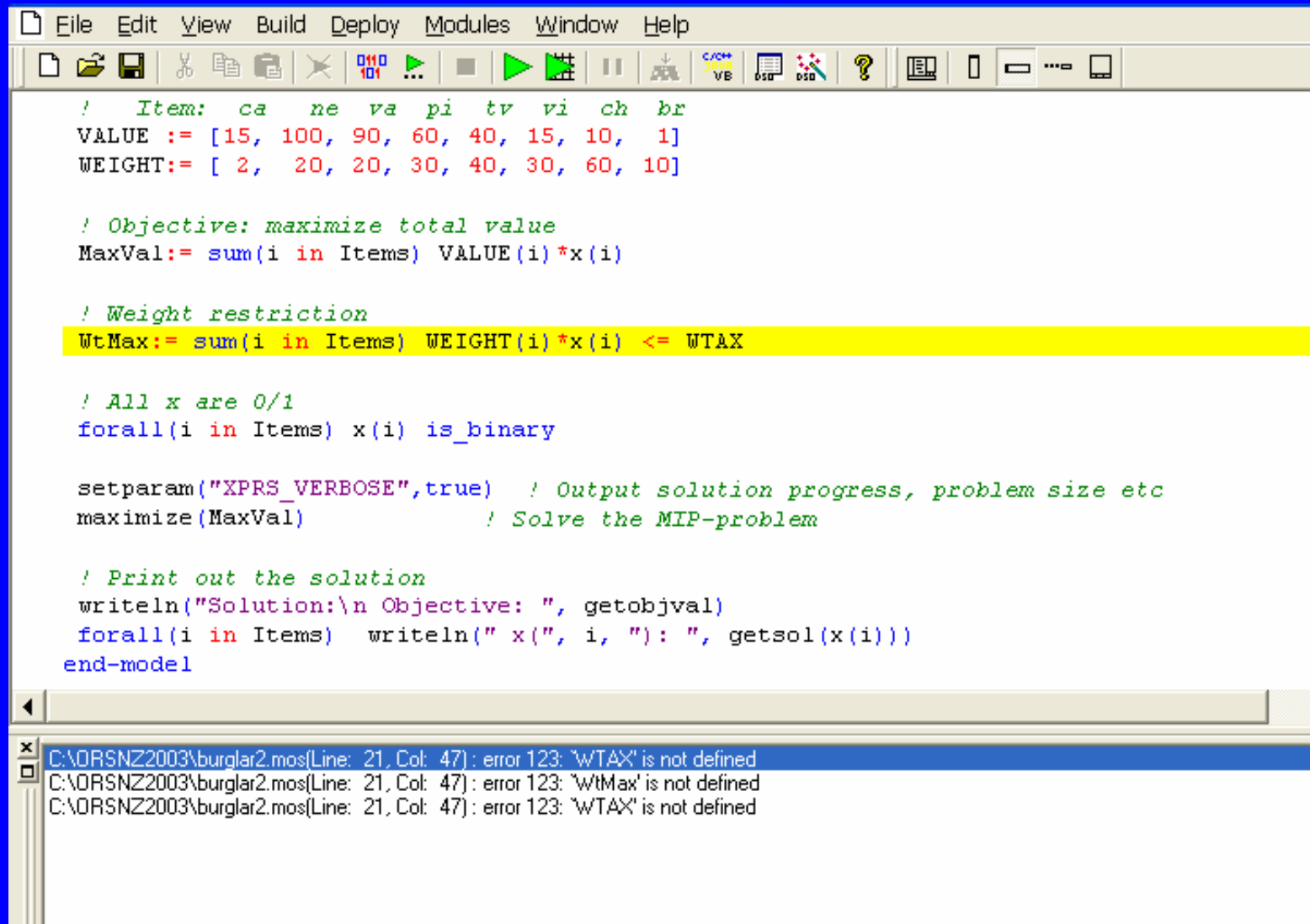
- "camera"
- "necklace"
- "vase"
- "picture"
- "tv"
- "video"
- "chest"
- "brick"

The dialog box has an 'OK' button at the bottom.

The background window also shows the following code:

```
writein("Solution:\n Objective: ", getobjval)
forall(i in Items) writein(" x(", i, "): ", getsol(x(i)))
end-model
```

# Syntax Error



The screenshot shows a GAMS IDE window with a menu bar (File, Edit, View, Build, Deploy, Modules, Window, Help) and a toolbar. The main editor displays a GAMS script for a knapsack problem. The script defines items (ca, ne, va, pi, tv, vi, ch, br) with their values and weights, sets an objective to maximize total value, and includes a weight restriction. The line `WtMax:= sum(i in Items) WEIGHT(i)*x(i) <= WTAX` is highlighted in yellow. Below this, there are commands for setting parameters, solving the problem, and printing the solution. The status bar at the bottom shows three error messages: `C:\ORSNZ2003\burglar2.mos(Line: 21, Col: 47): error 123: 'WTAX' is not defined`, `C:\ORSNZ2003\burglar2.mos(Line: 21, Col: 47): error 123: 'WtMax' is not defined`, and `C:\ORSNZ2003\burglar2.mos(Line: 21, Col: 47): error 123: 'WTAX' is not defined`.

```
! Item:  ca  ne  va  pi  tv  vi  ch  br
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)*x(i)

! Weight restriction
WtMax:= sum(i in Items) WEIGHT(i)*x(i) <= WTAX

! All x are 0/1
forall(i in Items) x(i) is_binary

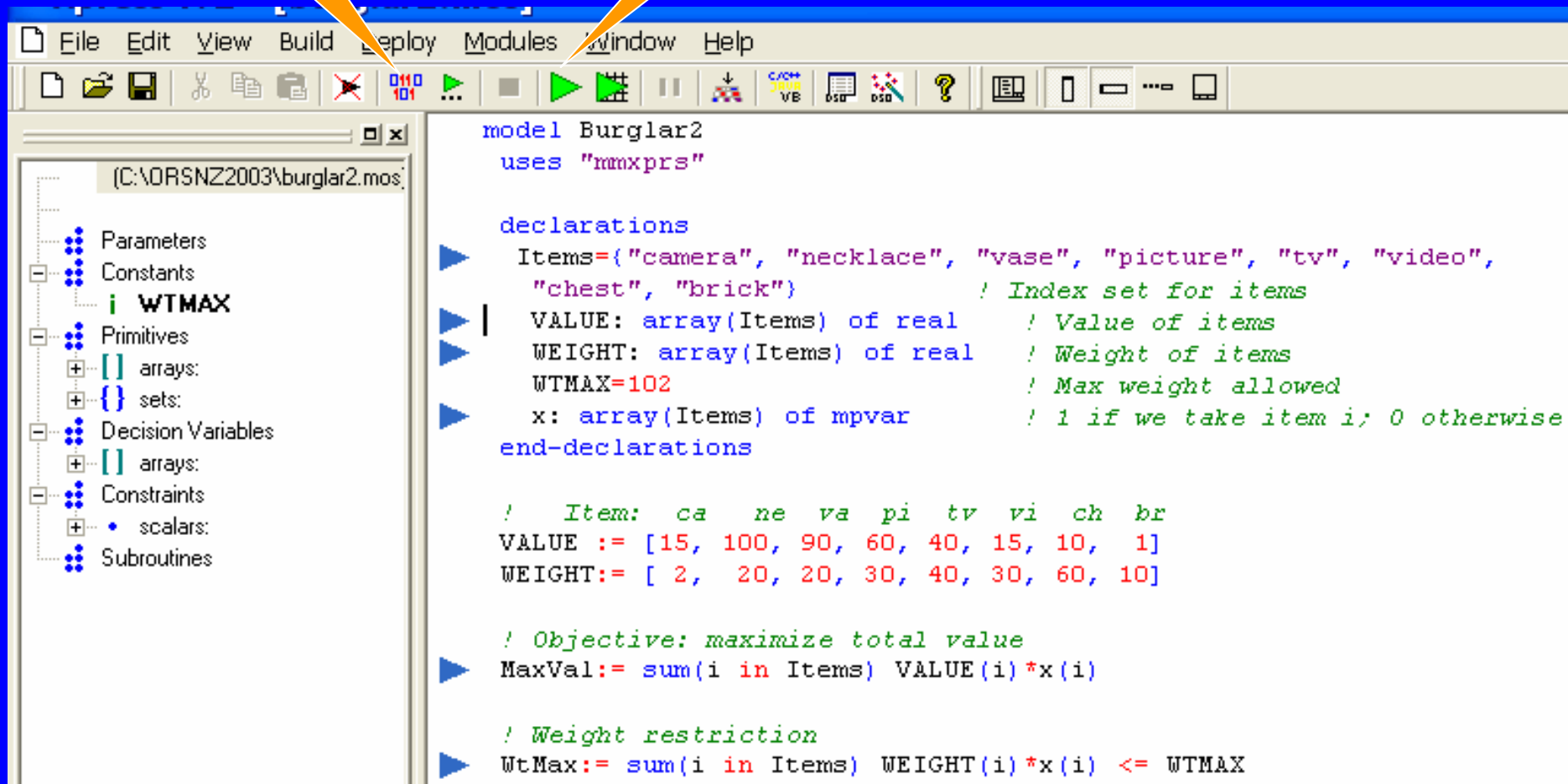
setparam("XPRS_VERBOSE",true)  ! Output solution progress, problem size etc
maximize(MaxVal)                ! Solve the MIP-problem

! Print out the solution
writeln("Solution:\n Objective: ", getobjval)
forall(i in Items)  writeln(" x(", i, "): ", getsol(x(i)))
end-model
```

C:\ORSNZ2003\burglar2.mos(Line: 21, Col: 47): error 123: 'WTAX' is not defined  
C:\ORSNZ2003\burglar2.mos(Line: 21, Col: 47): error 123: 'WtMax' is not defined  
C:\ORSNZ2003\burglar2.mos(Line: 21, Col: 47): error 123: 'WTAX' is not defined

Compile

Solve



The screenshot shows the GAMS (General Algebraic Modeling System) software interface. The left pane displays a project tree for the file [C:\VORSNZ2003\burglar2.mos]. The tree includes sections for Parameters, Constants (with a sub-entry 'WTMAX'), Primitives (with sub-entries for arrays and sets), Decision Variables (with a sub-entry for arrays), Constraints (with a sub-entry for scalars), and Subroutines. The main window displays the GAMS model code for 'Burglar2'. The code includes a 'uses' statement for 'mmxprs', a 'declarations' section defining items, value arrays, weight arrays, and a maximum weight constant, and an 'end-declarations' section. It also includes data for item values and weights, an objective function to maximize total value, and a weight restriction constraint.

```
model Burglar2
  uses "mmxprs"

  declarations
    Items={"camera", "necklace", "vase", "picture", "tv", "video",
           "chest", "brick"}           ! Index set for items
    VALUE: array(Items) of real        ! Value of items
    WEIGHT: array(Items) of real       ! Weight of items
    WTMAX=102                          ! Max weight allowed
    x: array(Items) of mpvar           ! 1 if we take item i; 0 otherwise
  end-declarations

  ! Item:  ca  ne  va  pi  tv  vi  ch  br
  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)*x(i)

  ! Weight restriction
  WtMax:= sum(i in Items) WEIGHT(i)*x(i) <= WTMAX
```

Reading Problem xprs\_e24\_13eae68

# Problem Statistics

1 ( 108 spare) rows  
 8 ( 0 spare) structural columns  
 8 ( 3432 spare) non-zero elements

# Global Statistics

8 entities 0 sets 0 set members

Presolved problem has: 1 rows 7 cols 7 non-zeros

Its	Obj Value	S	Ninf	Mneg	Sum Inf	Time
0	15.000000	D	0	0	.000000	0
1	295.000000	D	0	0	.000000	0

Optimal solution found

# Generating cuts

Its	Obj Value	Type	Cuts Added / Deleted	Time
1	280.000000	K	1 0	0

Cuts in the matrix : 1

Cut elements in the matrix : 5

Cuts in the cutpool : 1

Cut elements in the cutpool: 5

Node	Sols	Best Solution	Best Bound	Active	Time
* 1	1	280.000000	280.000000	0	0 *

\*\*\* Search completed \*\*\* Time: 0 Nodes: 1

Number of integer feasible solutions found is 1

Best integer solution found is 280.000000

# Solution:

Objective: 280

x(camera): 1

x(necklace): 1

x(vase): 1

x(picture): 1

x(tv): 0

x(video): 1

x(chest): 0

x(brick): 0

# Solving Within a Loop

```
maximize(MaxVal)
forall(loot in Items) do
    Fixit:= x(loot)=1
    maximize(MaxVal)
    writeln("\nMust steal ", loot, " Objective: ",getobjval)
    forall(i in Items | getsol(x(i))=1 ) write(" ",Items(i))
end-do
```

```
maximize(XPRS_LIN,MaxVal)
forall (loot in Items) write(loot,": ",getsol(x(loot))," ")
writeln("\nValue of a stronger sack = ",getdual(WtMax))
```

# Output from Example

Objective: 280

Must steal camera Objective: 280 camera necklace vase picture video

.....

Must steal tv Objective: 246 camera necklace vase tv brick

.....

Must steal chest Objective: 215 camera necklace vase chest

Optimal values of variables from MIP:

camera: 1 necklace: 1 vase: 1 picture: 1 tv: 0 video: 0 chest: 0 brick: 1

Solve as a relaxed LP only Objective: 286

camera: 1 necklace: 1 vase: 1 picture: 1 tv: 0.5 video: 0 chest: 0 brick: 1

Value of a stronger sack = 1



