Optimization Modeling and Programming in Xpress-Mosel

Alkis Vazacopoulos
CAPD Meeting
Carnegie Mellon University
Pittsburgh, PA
March 11-13, 2007
• Modeling Basics
  – Modeling and Programming Features
  – Optimization
  – Deployment

• Modeling Advanced
  – Algorithms
  – Debugging & Profiling
  – Mosel Modules
  – I/O Drivers
  – Model Separation
Developing an optimization model in Mosel

- Describe the business problem in the Mosel algebraic modeling language
- Ex: Capital Budgeting
  - Set of projects with net return
  - Each project has capital and personnel requirements
  - Limited capital and personnel resources
  - Select set of projects to maximize return
Developing an optimization model

Building Block 1

- Model name and parameters

```plaintext
model "CapitalBudgeting"
    uses "mmxprs" ! Use Xpress optimizer

    parameters ! Change at run-time
        DATA_FILE = ‘projects.dat’
    end-parameters

end-model
```
Developing an optimization model

Building Block 2

- Data declaration and reading

  declarations

  PROJECTS : set of string
  CAPITAL_MAX : real
  CAPITAL : array (PROJECTS) of real

  end-declarations

  initializations from DATA_FILE

  CAPITAL_MAX
  CAPITAL as 'CAPITAL_DATA'

  end-initializations
Developing an optimization model

Building Block 3

• Variable and constraint declarations

declarations

do_project: array( PROJECTS ) of mpvar
  ! Decision variable
MaxReturn, Capital, Personnel: linctr
  ! Constraints

end-declarations

forall ( p in PROJECTS )

do_project (p) is_binary
  ! Binary variable

 dash optimization
Developing an optimization model

Building Block 4

- Build constraints and objective

Total return is sum of selected projects returns

\[
\text{MaxReturn} := \sum_{p \text{ in } \text{PROJECTS}} \text{RETURN}(p) \times \text{do_project}(p)
\]
Developing an optimization model

Building Block 5
• Solve optimization model

\[ \text{maximize} \ (\text{MaxReturn}) \]
Developing an optimization model

Building Block 6

• Solution

if getprobstat=XPRS_OPT then
  writeln("Solution:
  Objective: ", getobjval)
  forall(i in PROJECTS)
    write(" x(" , i , "): ", getsol(x(i)))
end-if
Deploying Optimization Models

- Mosel source file
  - developer
- Mosel binary file
  - end-user

- protects intellectual property
- platform independent
- efficient
More Mosel Features

- **Modeling**
  - Variable: free, integer, partial integer, semi-continuous, sos1, sos2

- **Programming**
  - Dynamic arrays
  - Sparse data
  - Selections: if-then-elif-then-else, case
  - Loops: forall, while, repeat-until
Subroutines

Functions

```plaintext
forward function add ( a : integer, b : integer ) :
    integer

    
    Temp := add(10,20) ! this will return "30"

    
    function add ( a : integer, b : integer ) : integer

    returned := a + b

end-function
```

dash optimization
Algorithms

These functionalities allow

- Multiple optimization calls in one model
- Create and solve different problems in one model file
- Implement advanced algorithms, experiment and try more new ideas

**Benefit:** Spend more time in ‘designing’ rather than ‘implementing’
Algorithms: Modifying the Problem

After solving first problem, one can
- Create ‘additional’ new variables and constraints
- Delete existing constraints
- Add / delete variables to existing constraints
- Hide / Un-hide constraints

Solve second modified problem ....
Algorithms: Modifying the Problem

Add / delete variables to existing constraints

Constraint := 5*x + 2*y <= 20
...
maximize ( Objective_1 )
...
Constraint += 7*y
...
maximize ( Objective_2 )
...

the ‘revised’ constraint is

Constraint := 5*x + 9*y <= 20

dash optimization
Algorithms: Modifying the Problem

Can implement algorithms / heuristics such as

- Chronological decomposition of planning period in scheduling problems
- Draw efficient frontier by changing R.H.S.
- Add constraints and monitor change in objective
- Column Generation (Master and Sub-problem)
Algorithms

- Intermediate MIP solution
  Set-up for integer solution

  setcallback (XPRS_CB_INTSOL, "WriteReport")

  ! Callback to function ‘WriteReport’

  ....

  minimize (Cost)
Branch and Cut

Set-up for cut generation

```csharp
setcallback ( XPRS_CB_CM,
              "add_cut" )

! Cut manager callback to function ‘add_cut’

.......   

minimize ( Cost )
```
Debugging & Profiling

- Re/Set Breakpoint F9
- Set/Remove breakpoint condition
- Start / Continue F5
- Step over F10
- Step into F11
- Run to cursor F12
- Debug Options…
- Profile

- Re/Set Breakpoint F9
- Set/Remove breakpoint condition
- Start / Continue F5
- Step over F10
- Step into F11
- Run to cursor F12
- Debug Options…
- Profile

dash optimization
Modules

- mmodbc: ODBC connectivity
- mmquad: QP, MIQP
- mmxslp: NLP, MINLP
- mmssp: Stochastic LP, MILP
- mmive: Graphing
- mmxad: GUI builder
- kalis: CP

Extend Mosel: NI can be used to create user defined modules
Stochastic model

- mmrng: Simulation runs
- mmsp: Stochastic LP
- mmxad: Visual Interface
I/O drivers

- Data exchange between concurrent models:
  - reading and writing data from/to memory
  - synchronization of data access
  1. shmem
  2. mempipe
Model Separation

- module: mmjobs
- Sequential model
- Parallel model
- Exs:
  - Column Generation
  - Dantzig-Wolfe Decomposition
Dantzig-Wolfe Decomposition

- Multi-item, multi-period production planning
Dantzig-Wolfe decomposition

- Master Problem
- Sub Problem for Factory 1
- Sub Problem for Factory 2
- Proposals
Modeling statements with Xpress-Mosel and Xpress-IVE

```
declarations
    DemandCities: set of string
    SupplyCities: set of string

    TCOST : array(SupplyCities, DemandCities) of real
    DIST: array(SupplyCities, DemandCities) of real
    DEMAND: array(DemandCities) of integer

    'Decision variables
    x : array(SupplyCities, DemandCities) of mpvar /flow fraction
    y : array(SupplyCities) of mpvar /1 open, 0 closed

end-declarations

!constraints

forall i in SupplyCities | y(i) is binary

TotalCost := sum(j in DemandCities) TCOST(i,j) * DEMAND(j) * x(i,j)

!limit on number of open facilities
MaxN := sum[j in SupplyCities] | 'Shortfall' | y(j) <= N

!ship only if facility is open
forall i in SupplyCities, j in DemandCities | R_CUTS(i,j) := x(i,j) <= y(j)

!satisfy demand
forall j in DemandCities
R_DEM[j] := sum(i in SupplyCities | exists(DIST(i,j) and DIST(i,j) < SRW) x(i,j) = 1

!call optimizer
minimize(TotalCost)
```

dash optimization
Modeling and programming statements with Xpress-Mosel

```plaintext
! constraints
forall(i in SupplyCities) y[i] is_binary

TotalCost := sum(i in SupplyCities, j in DemandCities) TCOST(i, j) * DEMAND(j) * x(i, j)

! ship only if facility is open
forall(i in SupplyCities, j in DemandCities) R_CUTS(1, j) := x(i, j) <= y[i]

! satisfy demand
forall(j in DemandCities)
  RMIN[j] := sum(i in SupplyCities) exist(DIST(1, i)) and DIST(1, i) <= SERV) x(1, i) = 1

/***************
! solve for different values of the limit on number of open facilities
forall(n in 1..12) do
  ! limit on number of open facilities
  MaxN := sum(i in SupplyCities) x('Shortfall'[i][j][n]) <= n

! call optimizer
minimize(TotalCost)

! call reporting procedure
report_solution
end-do

***************
```
Browsing the solution and model entities in Xpress-IVE
Run statistics in Xpress-IVE

### Matrix
- Rows (constraints): 8758
- Columns (variables): 3955
- Nonzero elements: 17453
- Global entities: 145
- Sets: 0
- Set members: 0

### Presolved
- Rows (constraints): 1389
- Columns (variables): 1332
- Nonzero elements: 4849
- Global entities: 139
- Sets: 0
- Set members: 0

### Overall status
- Finished global search

### LP relaxation
- Algorithm: Simplex dual
- Simplex iterations: 2001
- Objective: 7.2e+005
- Status: LP Optimal
- Time: 0.5s

### Global search
- Current node: 1
- Depth: 0
- Active nodes: 0
- Best bound: 7.2e+006
- Best solution: 7.2e+006
- Gap: 0%
- Status: Solution is optimal
- Time: 0.5s
Problem matrix in Xpress-IVE
Problem matrix/solution in Xpress-IVE
Deployment wizard in Xpress-IVE

The candidate file for deployment is:
C:\location20compete\locdemo01.mos

How would you like to use this Mosel model in your application?

Save .BIM file
- With debug info
- All names stripped

Run Mosel model from
- C
- Java
- Visual Basic
- VB.NET
- C#

Optimize matrix file from
- C
- Java
- Visual Basic
- VB.NET
- C#

To directly create a Windows executable that runs a .BIM file:
1. Copy C:\XpressMP\bin\mrun.exe to the same folder as the .BIM file;
2. Rename mrun.exe to match the name of the .BIM file, but with .EXE instead.

Next > Cancel
Deployment wizard in Xpress-IVE

Function run_mosel(ByRef err As Long) As Long
Dim result As Long
Dim model As Long

err = XPRMinit
If err <> 0 And err <> 32 Then
er = 1
run_mosel = 0
Exit Function
End If

model = XPRMloadmod("c:\location20complete\locdemo01.bim", ",")
If model = 0 Then
err = 2
run_mosel = 0
Exit Function
End If
Visualization, model controls, and mapping with Xpress-XAD
One Program: Mosel model as procedure called from GUI

```mosel
procedure run_model|m:integer, switch_silo:boolean,
  forceopen:array(string) of boolean, forceclosed:array(string) of boolean,
  service_realm:switch, switch_hour:boolean, use_capacity:boolean, use_fixedcost:boolean,
  switch_type:integer, budget:real,

  if not use_fixedcost then
    FixedCost := FCOST("Shortfall") * y("Shortfall")
  else
    FixedCost := sum (i in SupplyCities) FCOST(i) * y(i)
  end-if

  VariableCost := sum (i in SupplyCities, j in DemandCities) TCOST(i,j) * DEMAND[j] * x(i,j)
  TotalCost := FixedCost + VariableCost

  FixedCostR := FixedCost - FCOST("Shortfall") * y("Shortfall")
  VariableCostR := sum (i in SupplyCities, j in DemandCities) TCOST(i,j) * DEMAND[j] * x(i,j)
  TotalCostR := FixedCostR + VariableCostR

  TotalFacilities := sum (i in SupplyCities) y("Shortfall")
  TotalDemServed := sum (i in SupplyCities, j in DemandCities) DEMAND[j] * x(i,j)
  DistDemServed := sum (i in SupplyCities, j in DemandCities) DEMAND[j] * x(i,j) / DIST(i,j)

if not switch_silo then
  MaxN := sum (i in SupplyCities) y("Shortfall") * y(i) <= n
else
  MaxN := 0
end-if
```
Interactive solving through GUI
with all data in computer memory
Scenario Management

US locations

Solution types
- Minimize cost
- Minimize DCs
- Maximize Coverage
- Coverage /Budget

Characteristics
- Single sourcing
- Use capacity
- Use fixed costs

Number of DCs
- Up to: 8
- Model decides

Service: 100%

Minimize cost

Max N=8; Multiple sourcing; Service percentage=100%; Sol Type: 1
Max N=8; Single sourcing; Service percentage=100%; Use capacity; Sol Type: 1
Max N=8; Single sourcing; Service percentage=100%; Use fixed costs; Sol Type: 1
Max N=8; Multiple sourcing; Service percentage=80%; Sol Type: 1
Max N=8; Multiple sourcing; Service percentage=100%; Sol Type: 1CLOSED (Reading, PA);
Max N=8; Multiple sourcing; Service percentage=100%; Sol Type: 1OPEN (Charleston, WV); CLOSED (Reading, PA);
Max N=8; Multiple sourcing; Service percentage=70%; Use capacity; Sol Type: 1

Number DCs: 8
Demand Served: 100
Service Avg/Max: 238/500
Total cost: $1,962,968
Fixed cost: $0
Variable cost: $1,962,968

dash optimization
Scenario comparison/visualization
Scenario with stricter service miles requirements and shortfalls
Visualization: Pop-up information
Visualization: Forcing facilities open or closed