
dash optimization

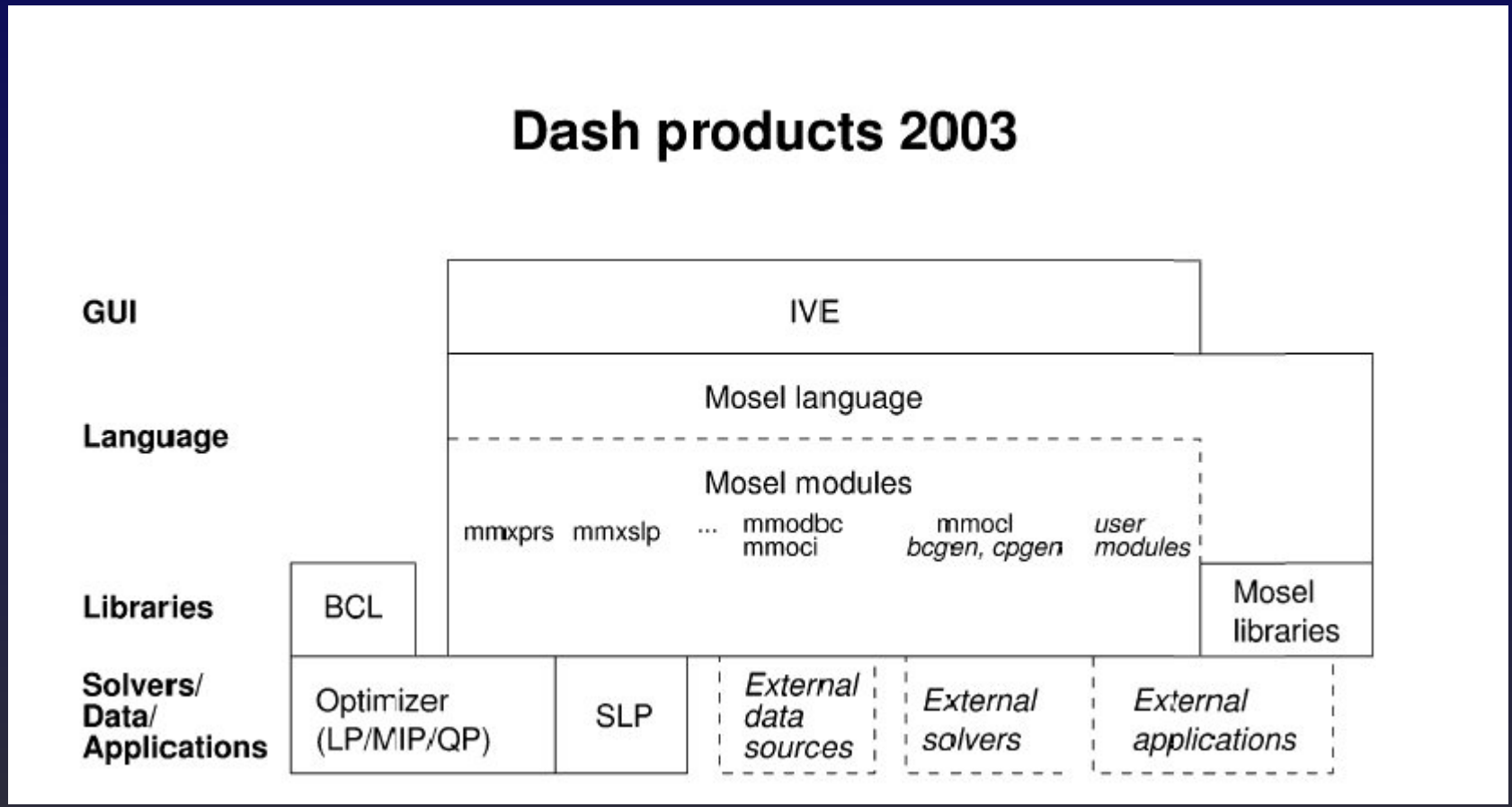
Xpress-Mosel

Bob Daniel
bob.daniel@dashoptimization.com
www.dashoptimization.com

Xpress-MP: Mosel's Context

- Overview
- Optimizer
- BCL
- Mosel

Dash products



Xpress-Optimizer

- LP solver
  - primal and dual simplex
  - Newton Barrier
- MIP solver
- QP and MIQP solver
- SLP solver for NLP; MISLP solver for MINLP

Xpress-BCL

- Library for modeling and basic solution tasks
  - model LP, MIP, QP, MIQP problems
  - retrieve solution information for model objects
  - work with multiple problems
  - automatic matrix regeneration
  - interfaceable to external software
  - use of all Optimizer functions possible
  - C, C++, Java, and VB interfaces

BCL: Modeling Functions

- Problem handling
- Variables
- Constraints, linear and quadratic expressions
- Special Ordered Sets and other IP objects
- Data input and index sets

BCL: Example

- Please refer to the handout

Xpress-Mosel

- A modeling and solving environment
  - Integration of modeling and solving
  - Programming facilities
  - Open, modular architecture
- Book:
  "Applications of optimization with Xpress-MP"
  Christelle Guéret, Christian Prins & Marc Sevaux
  Translated and revised by Susanne Heipcke
  Dash Optimization, 2002, ISBN 0-9543503-0-8

Components and Interfaces

- Language
  - implement problems and solution algorithms by Model or Mosel program
- Model Compiler & Run-time Libraries
  - compile, execute and access models from e.g. C, C++, Java, VB or Delphi
- Mosel Native Interface (NI)
  - to provide new or extend existing functionality of the Mosel language (modules)
- Xpress-IVE
  - GUI, representation of the problem matrix, solution status/progress graphs, and result display

Mosel Language

- Decision variables, linear constraints
- Arrays, (index-)sets
- Aggregate operators: sum, prod, min, max, and, or, union, intersection
- Matrix generator and export (LP+MPS)
- Data initialization from files

First Mosel Example

- Please refer to the handout

Language Features

- Data structures: array, set
- Selections: if-then-[elif-then]-[else], case
- Loops: forall-[do], while-[do], repeat-until
- Operators:
  - standard arithmetic operators
  - aggregate operators (sum, prod, and, or, min, max, union, intersection)
  - set operators
- Subroutines: functions, procedures
  - forward declaration, overloading

Programming: Primes (1/2)

```
model "Prime by Sieve"
 parameters            ! Can be changed at execution time
  LIMIT=100
 end-parameters
 declarations
  SNumbers, SPrime: set of integer
  n,i: integer
 end-declarations
 Snumbers := {2..LIMIT}
```

Programming: Primes (2/2)

```
writeln("Prime numbers between 2 and ", LIMIT, ":")
n:=2
repeat
  while (not(n in SNumbers)) n+=1
  SPrime += {n}
  i:=n
  while (i<=LIMIT) do
    SNumbers -= {i}
    i += n
  end-do
until SNumbers={}
writeln(SPrime)
end-model
```

A Solving Environment

- No separation between modeling statements and solving statements
- Programming facilities for pre/postprocessing, algorithms
- Principle of incrementality
- Not solver-specific
- Ability to interact with solver(s)

Solving: Variable Fixing Heuristic

- Solution heuristic written with Mosel
- Program split into several source files
- Please refer to the handout

Mosel Libraries

- Model Compiler Library
  - compile to a virtual machine
  - binary format is architecture independent
- Runtime Library
  - load and run binary (models)
  - access to Mosel internal database (data, solution values, ...)

Mosel: A Modular Environment

- Open architecture
- Module = dynamic libraries
- Not dedicated to any particular use: e.g.
  - solvers: Xpress-Optimizer(LP, MIP, QP, SLP), CHIP, OptQuest
  - database access: ODBC, OCI
  - system commands

Modules

- Dynamic library written in C that observes the conventions of the Mosel Native Interface
- Modules extend the Mosel language with
  - constant symbols
  - subroutines
  - types
  - control parameters

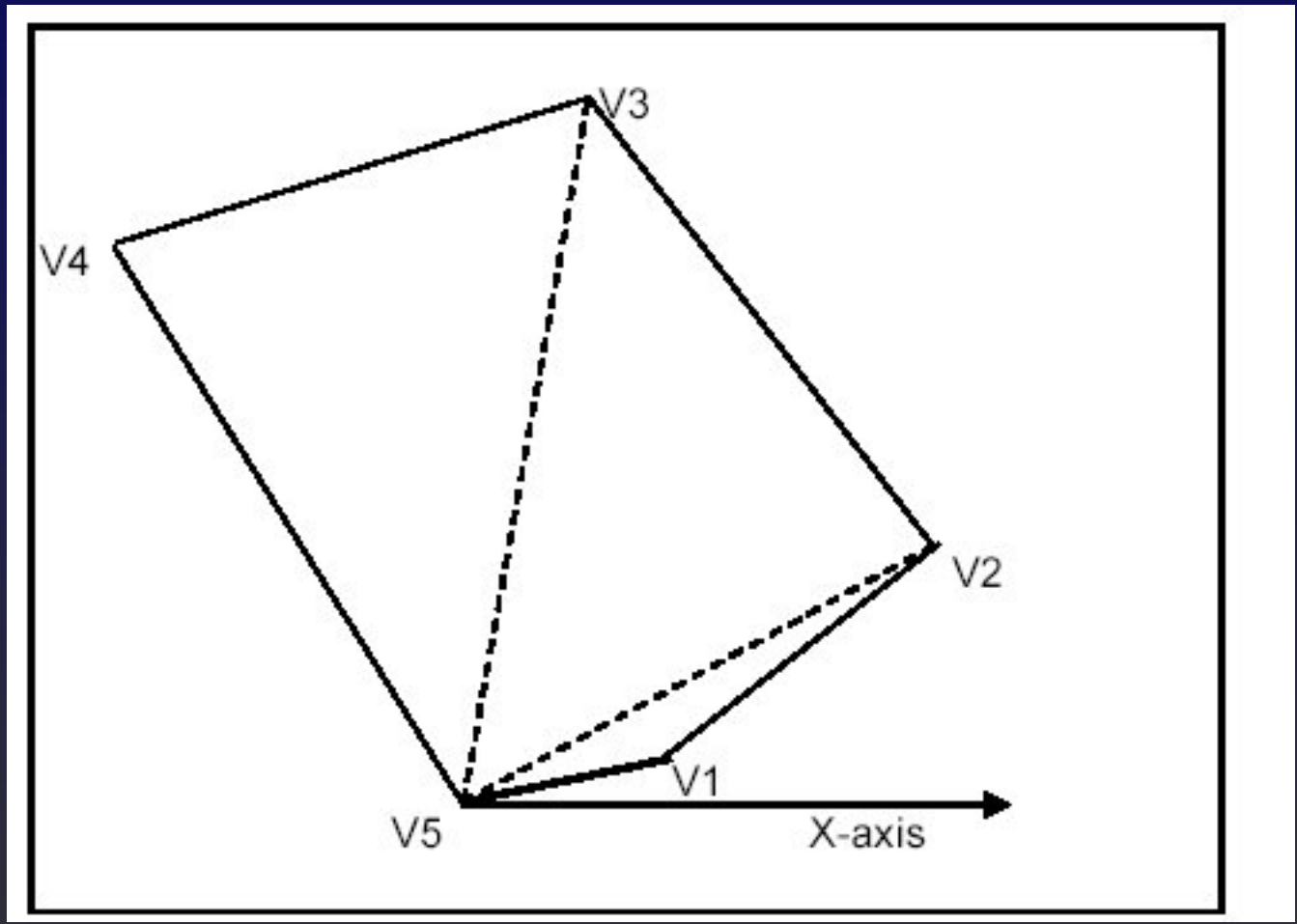
mmxprs: using Optimizer callbacks

```
uses "mmxprs"
(! Implements Xpress' User Integer Solution callback function, called
   every time an integer solution is found during the Branch and Bound
   search. !)

declarations
 x: array(1..10) of mpvar
end-declarations

procedure UIS
 writeln("Solution:", getsol(Objective))
 forall(i in 1..10)
   write("x(", i, ")=", getsol(x(i)), "\t")
 writeln
end-procedure

setcallback(XPRS_CB_UIS, "UIS")
```

mmxslp: NLPs by SLP (1/4)

- Problem definition
  - The diameter of a two-dimensional shape is the greatest distance between any two of its points
    - for a circle, this definition corresponds to the normal meaning of diameter
    - for a polygon (with straight sides), it is equivalent to the greatest distance between any two vertices
  - What is the greatest area of a polygon with N sides and a diameter of 1?

mmxslp: NLPs by SLP (2/4)



mmxslp: NLPs by SLP (3/4)

- Vertex V_N chosen as "base" point
- Other vertices are measured from it, using (r,θ) coordinates – distance (r) is measured from the vertex, and the angle of the vertex (θ) is measured from the X-axis
  - (r_i, θ_i) are coordinates of vertex V_i
  - Area of the triangle $V_N V_i V_j = \frac{1}{2} * r_i * r_j * \sin(\theta_j - \theta_i)$
  - Side $V_i V_j$ is given by: $(V_i V_j)^2 = r_i^2 + r_j^2 - 2 r_i r_j \cos(\theta_j - \theta_i)$
  - Total area of polygon is sum(i=2..N-1) area($V_N V_i V_{j-1}$)
  - Maximum diameter being 1 requires that all the sides of all the triangles are ≤ 1, i.e. $r_i \le 1$ for i=1..N-1 and $V_i V_j \le 1$ for i=1..N-2, j=i+1..N-1
  - Also insist vertices are ordered: $\theta_i \le \theta_i + 1$

mmxslp: NLPs by SLP (4/4)

- Please refer to the handout
- Demo

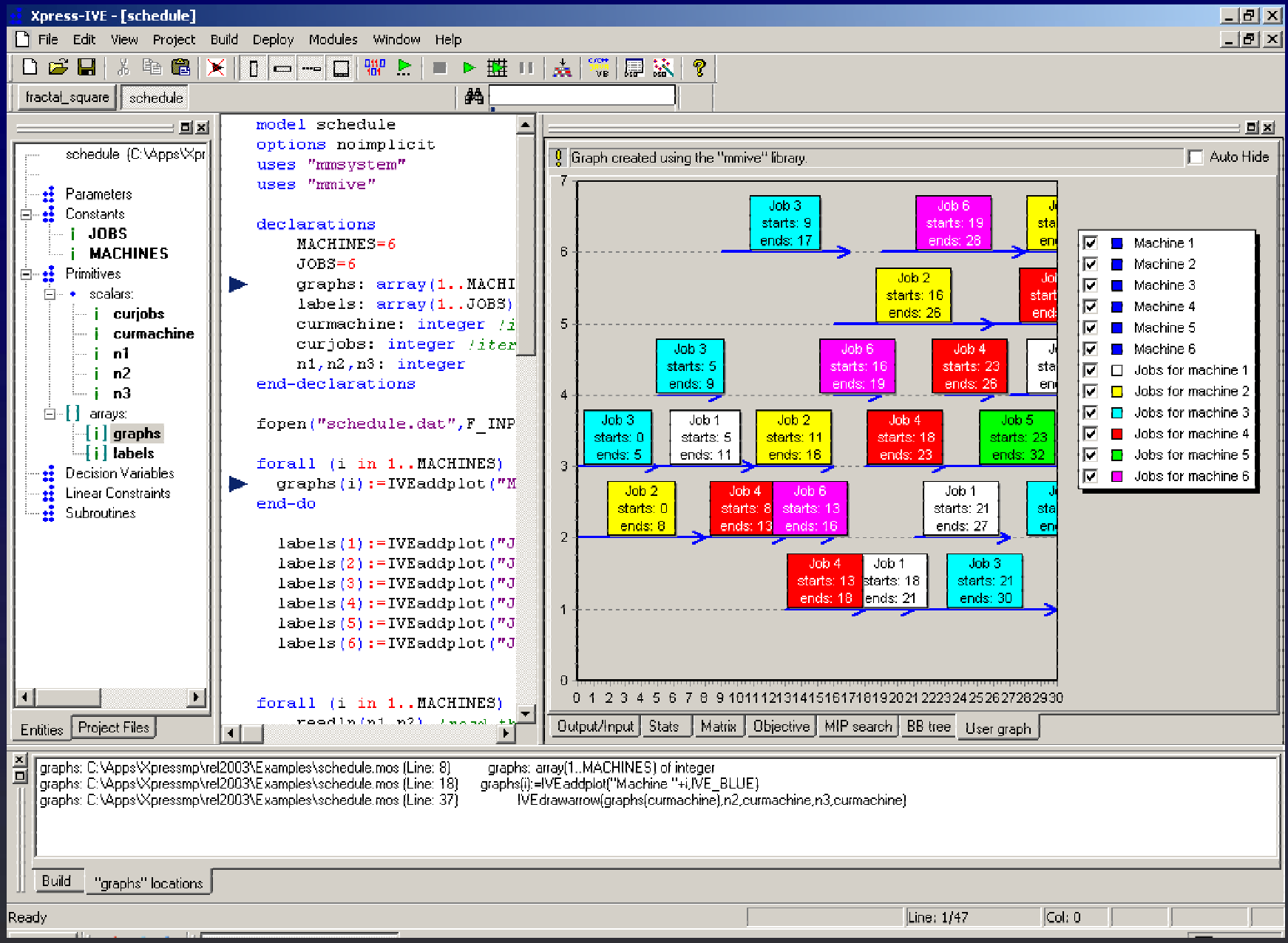
- Note the interface with Excel for e.g. process models

mmodbc: Using SQL

```
uses "mmodbc"
declarations
  Nprod, Nrm: integer
end-declarations
SQLconnect('DSN=MSExcel;DBQ=ssxmpl.xls')
Nprod := SQLreadinteger("select Nprod from SIZES")
Nrm   := SQLreadinteger("select Nrm   from SIZES")
declarations
  PneedsR: array(1..Nprod,1..Nrm) of real
end-declarations
SQLexecute("select * from USAGE", [PneedsR])
forall(p in 1..Nprod, r in 1..Nrm) do
  writeln("PneedsR(",p,",",r, " is ", PneedsR(p,r) )
end-do
end-model
```

mmive: User Graphs

- Please refer to the handout
- Possible output on next slide


Xpress-IVE - [schedule]
File Edit View Project Build Deploy Modules Window Help
fractal_square
schedule
schedule (C:\Apps\Xpr
Parameters
Constants
JOBS
MACHINES
Primitives
scalars:
curjobs
curmachine
n1
n2
n3
arrays:
graphs
labels
Decision Variables
Linear Constraints
Subroutines
Entities
Project Files
model schedule
options noimplicit
uses "mmsystem"
uses "mmive"
declarations
MACHINES=6
JOBS=6
graphs: array(1..MACHI
labels: array(1..JOBS)
curmachine: integer !i
curjobs: integer !iter
n1,n2,n3: integer
end-declarations
fopen("schedule.dat",F_INP
forall (i in 1..MACHINES)
graphs(i):=IVEaddplot("M
end-do
labels(1):=IVEaddplot("J
labels(2):=IVEaddplot("J
labels(3):=IVEaddplot("J
labels(4):=IVEaddplot("J
labels(5):=IVEaddplot("J
labels(6):=IVEaddplot("J
forall (i in 1..MACHINES)
Graph created using the "mmive" library.
Auto Hide
Job 3 starts: 9 ends: 17
Job 6 starts: 19 ends: 28
Job 2 starts: 16 ends: 25
Job 3 starts: 5 ends: 9
Job 6 starts: 16 ends: 19
Job 4 starts: 23 ends: 26
Job 3 starts: 0 ends: 5
Job 1 starts: 5 ends: 11
Job 2 starts: 11 ends: 16
Job 4 starts: 18 ends: 23
Job 5 starts: 23 ends: 32
Job 2 starts: 0 ends: 8
Job 4 starts: 8 ends: 13
Job 6 starts: 13 ends: 16
Job 1 starts: 21 ends: 27
Job 4 starts: 13 ends: 18
Job 1 starts: 18 ends: 21
Job 3 starts: 21 ends: 30
Machine 1
Machine 2
Machine 3
Machine 4
Machine 5
Machine 6
Jobs for machine 1
Jobs for machine 2
Jobs for machine 3
Jobs for machine 4
Jobs for machine 5
Jobs for machine 6
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
Output/Input
Stats
Matrix
Objective
MIP search
BB tree
User graph
graphs: C:\Apps\Xpressmp\rel2003\Examples\schedule.mos (Line: 8) graphs: array(1..MACHINES) of integer
graphs: C:\Apps\Xpressmp\rel2003\Examples\schedule.mos (Line: 18) graphs(i):=IVEaddplot("Machine "+i,IVE_BLUE)
graphs: C:\Apps\Xpressmp\rel2003\Examples\schedule.mos (Line: 37) IVEdrawarrow(graphs(curmachine),n2,curmachine,n3,curmachine)
Build
"graphs" locations
Ready
Line: 1/47
Col: 0

Mosel Native Interface

- Set of conventions for the definition of Mosel modules
- Functions for accessing the currently executed model (similar to Mosel Run-time Library)
- Can interact with and make changes to model objects (not possible with Mosel Run-time Library)

Module complex: Define a Type

```
uses "complex"
 declarations
  c: complex
  t: array(1..10) of complex
 end-declarations

 forall(j in 1..10) t(j):= complex(j, 10-j) ! Initialize complex
 t(5):= complex("5+5i")                     ! Assign

 c:= prod(i in 1..5) t(i) ! Aggregate PROD operator …
 writeln("test sum: ", sum(i in 1..10) t(i)) ! … and SUM

! Using arithmetic operators on complex objects
 c:= t(1)*t(3)/t(4) + if(t(2)=0, t(10), t(8)) + t(5) - t(9)

 initializations to "complex_test.dat"  ! File output of complex
  c t
 end-initializations
```

Mosel Tools

- mod2mos
  - mp-model to Mosel translator, to convert mp-model files into Mosel programs/models (incl. data files)
  - Comprehensive documentation of possible problems and recommendations on how to improve the performance of generated models
- Xpress-IVE
  - Graphical user interface for Mosel
  - Representation of the problem matrix, solution status/progress graphs, and result display

Module Wizard in Xpress-IVE

- IVE's wizard greatly eases building your own modules
- Just add C to implement your functionality. All interfacing done by wizard

Embedding Mosel: issues

- Speed
  - Don’t have to compile model each time
- Security
  - No release of intellectual property
  - Users can’t disrupt model

Embedding Mosel

```
/* Load compiled model */

#include <stdio.h>
#include "xprm_rt.h"

int main(int argc, char *argv[])
{
  XPRMmodel model;
  int result;

  XPRMinit();                          /* Initialize Mosel */
  model = XPRMloadmod("foliodata.bim", NULL);
  XPRMrunmod(model, &result, NULL);
  XPRMunloadmod(model);
  return 0;
}
```

Parameterized model execution

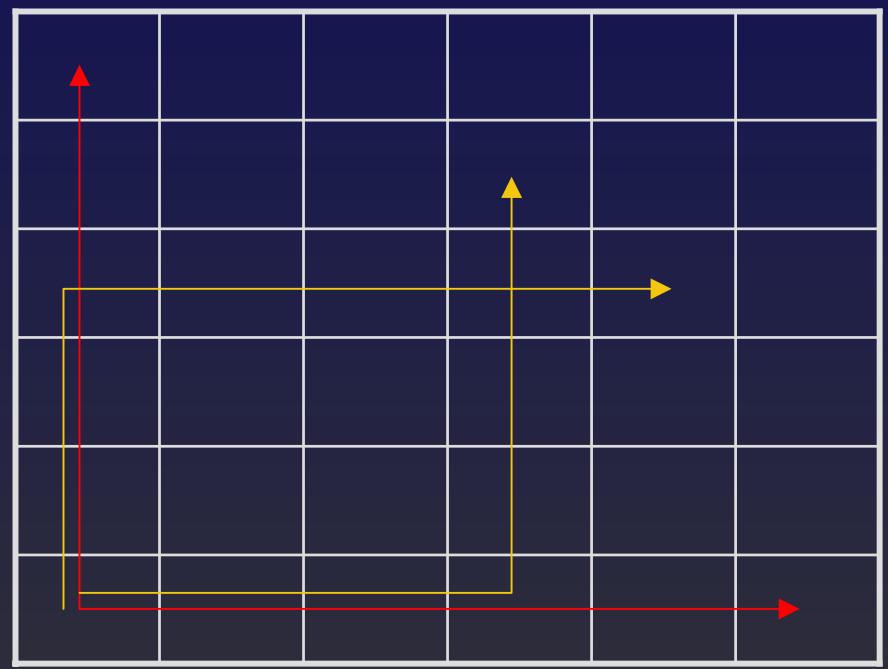
```
/* Execute = compile/load/run a model */
#include <stdio.h>
#include "xprm_mc.h"
int main(int argc, char *argv[])
{
  XPRMmodel model;
  int result;   char params[128];

  XPRMinit();                /* Initialize Mosel */

  sprintf(params,
   "OUTFILE=result2.dat,MAXRISK=0.4,MAXVAL=0.25");

  XPRMexecmod(NULL,"foliodata", params, &result, &model);

  XPRMunloadmod(model);
  return 0;
}
```

Accessing model results

```
#include <stdio.h>
#include "xprm_mc.h"

int main(int argc, char *argv[])
{
  XPRMmodel model;
  int result;

  XPRMinit();                  /* Initialize Mosel */
                               /* Execute = compile/load/run a model */
  XPRMexecmod(NULL, "foliodata", NULL, &result, &model);

                               /* Test whether a solution is found */
  if((XPRMgetprobstat(model)&XPRM_PBRES)==XPRM_PBOPT)
    printf("Objective value: %g\n", XPRMgetobjval(model));
  return 0;
}
```

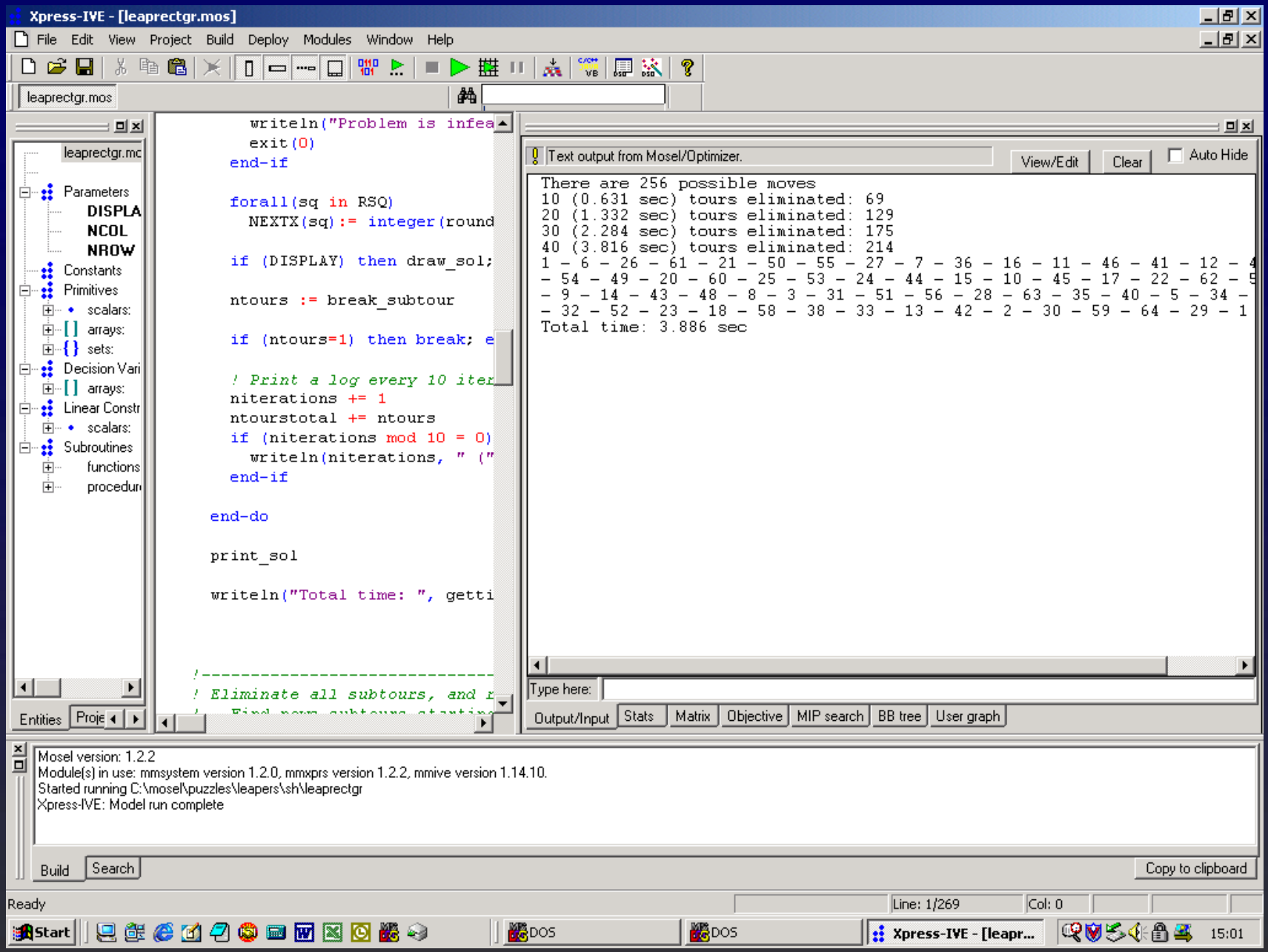
The 5-leaper knight

- Variant of regular chess knight
- Jumps 5 squares in a line, or 3 across and 4 up
- (x1-x2)^2 + (y1-y2)^2 = 25

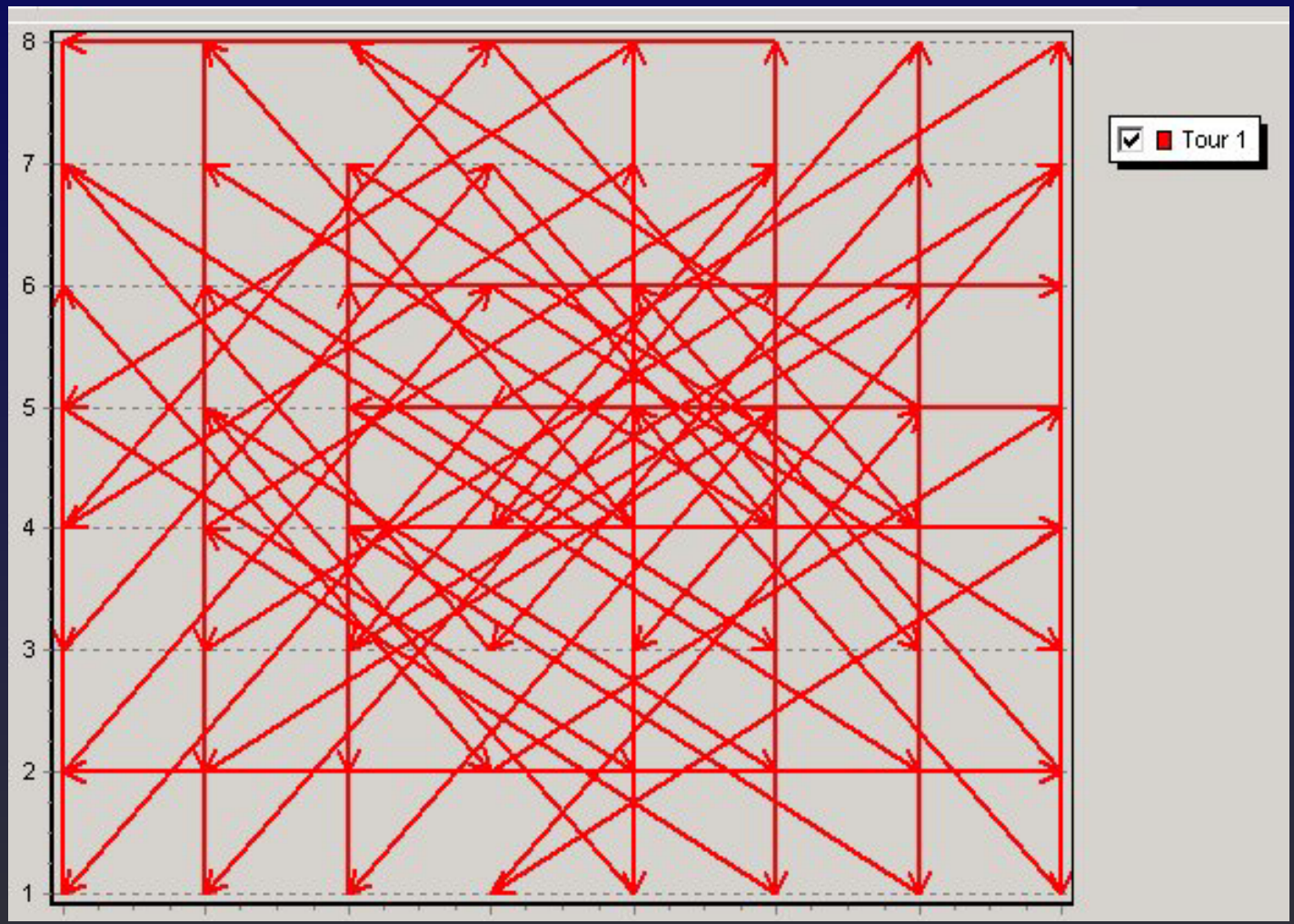


The 5-leaper knight

- Please refer to the handout
- Next slides are Xpress-IVE screens


Xpress-IVE - [leaprectgr.mos]
File Edit View Project Build Deploy Modules Window Help
leaprectgr.mos
writeln("Problem is infea
exit(0)
end-if
forall(sq in RSQ)
NEXTX(sq):= integer(round
if (DISPLAY) then draw_sol;
ntours := break_subtour
if (ntours=1) then break; e
! Print a log every 10 iter
niterations += 1
ntourstotal += ntours
if (niterations mod 10 = 0)
writeln(niterations, " ("
end-if
end-do
print_sol
writeln("Total time: ", getti
! Eliminate all subtours, and r
leaprectgr.mc
Parameters
DISPLA
NCOL
NROW
Constants
Primitives
scalars:
arrays:
sets:
Decision Vari
arrays:
Linear Constr
scalars:
Subroutines
functions
procedur
Entities
Proje
Text output from Mosel/Optimizer.
View/Edit
Clear
Auto Hide
There are 256 possible moves
10 (0.631 sec) tours eliminated: 69
20 (1.332 sec) tours eliminated: 129
30 (2.284 sec) tours eliminated: 175
40 (3.816 sec) tours eliminated: 214
1 - 6 - 26 - 61 - 21 - 50 - 55 - 27 - 7 - 36 - 16 - 11 - 46 - 41 - 12 -
- 54 - 49 - 20 - 60 - 25 - 53 - 24 - 44 - 15 - 10 - 45 - 17 - 22 - 62 -
- 9 - 14 - 43 - 48 - 8 - 3 - 31 - 51 - 56 - 28 - 63 - 35 - 40 - 5 - 34 -
- 32 - 52 - 23 - 18 - 58 - 38 - 33 - 13 - 42 - 2 - 30 - 59 - 64 - 29 - 1
Total time: 3.886 sec
Type here:
Output/Input
Stats
Matrix
Objective
MIP search
BB tree
User graph
Mosel version: 1.2.2
Module(s) in use: mmsystem version 1.2.0, mmxprs version 1.2.2, mmive version 1.14.10.
Started running C:\mosel\puzzles\leapers\sh\leaprectgr
Xpress-IVE: Model run complete
Build
Search
Copy to clipboard
Ready
Line: 1/269
Col: 0
Start
DOS
DOS
Xpress-IVE - [leapr...
15:01

8 x 8 5-leapers tour



Mosel: Summary

A modeling and solving environment

- Integration of modeling and solving
- Programming facilities
- Open, modular architecture


dash optimization