Introduction to Generated Math Programs

Marcel Hunting
AIMMS Optimization Specialist

Webinar, October 15, 2014

Overview

- Introduction
- GMP library
- Multiple solutions
- Solver sessions

INTRODUCTION

Solve statement – What happens?

```
Solve myMP;
```

1) aimms generates model
2) solver retrieves model from aimms
3) solver actually solves model
4) aimms retrieves solution from solver (and stores it)

Resolve model – What happens?

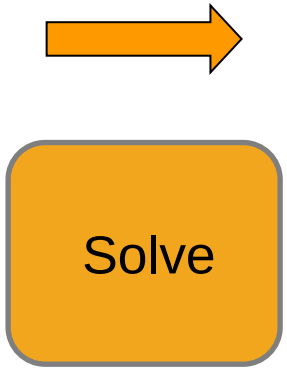
```
Solve myMP;
Change model data;
Solve myMP;
```

1) aimms *partially* or *completely* re-generates model (based on number of changes)
2) solver retrieves updates (or complete model) from aimms
3) and 4)

Model generation

Variables:
- JobSchedule(j,s)
- StartTime(s,m)
- TimeSpan

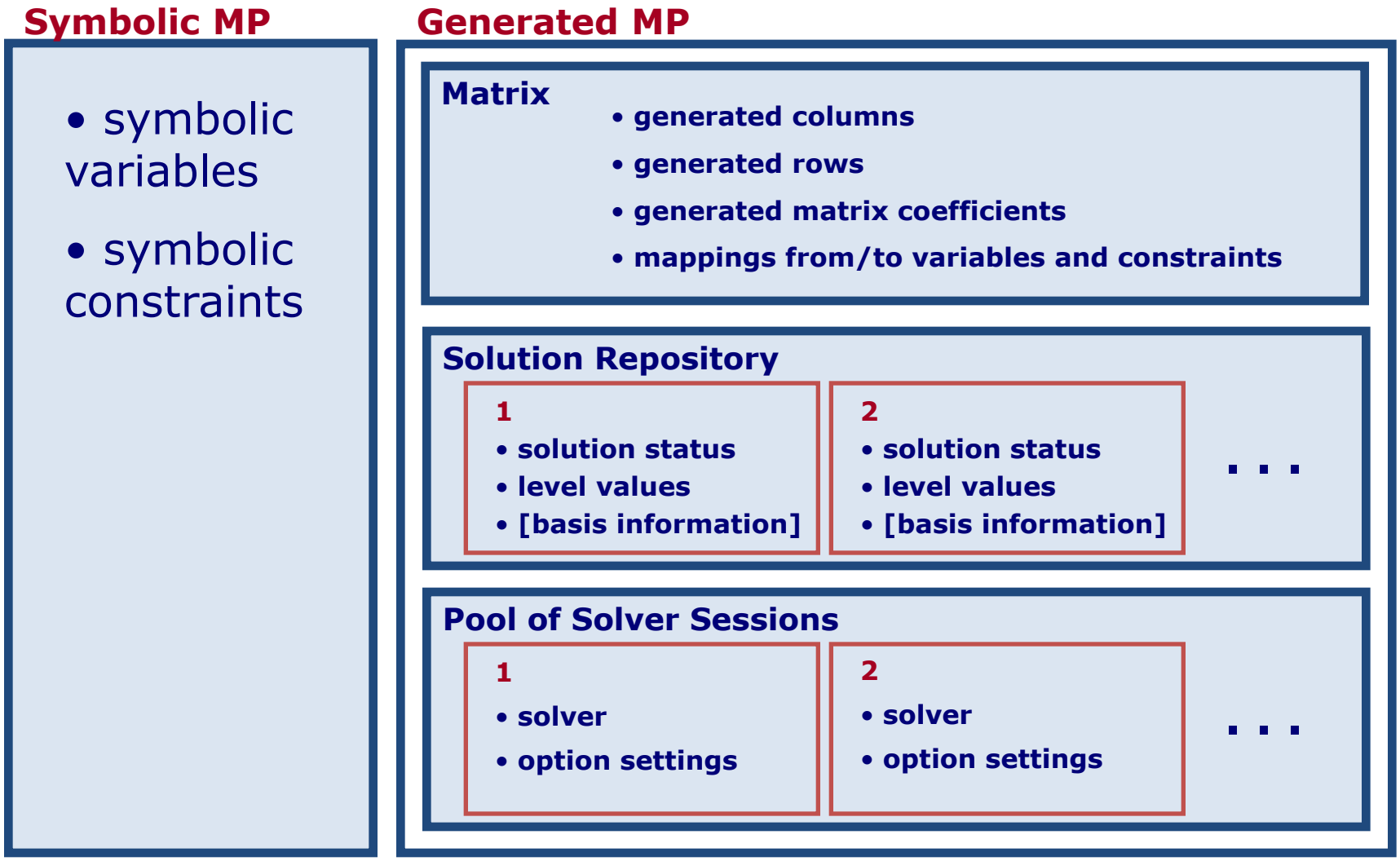
Constraints:
- OneJobPerSchedule(s)
- OneSchedulePerJob(j)
- MachineStartTime(s,m)
- ScheduleStartTime(s,m)



Columns:
- c0 … c48
- c49 … c118
- c119

Rows:
- r0 … r6
- r7 … r13
- r14 … r76
- r77 … r136

Generated Math Program (GMP)



Basic GMP

Normally:

```
solve MathProgram;
```

GMP:

```
myGMP := GMP::Instance::Generate(MathProgram);
GMP::Instance::Solve(myGMP);
```

Modify:

```
GMP::Column::SetUpperBound(myGMP,StartTime(s1,m1),5);
```

GMP LIBRARY

GMP namespaces

- GMP::Instance
- GMP::Column
- GMP::Coefficient
- GMP::Row
- GMP::Solution
- GMP::SolverSession
- GMP::Event
- GMP::Benders
- GMP::Linearization
- GMP::ProgressWindow
- GMP::Stochastic
- GMP::Tuning

GMP update functions

- GMP::Column::XXX(GMP, column, …)

  Add, Delete, Freeze, SetType, SetLowerBound, SetUpperBound, GetStatus, GetName

- GMP::Row::XXX(GMP, row, …)

  Add, Delete, Generate, SetRightHandSide, GetRightHandSide, GetName

- GMP::Coefficient::XXX(GMP, row, column, …)

  Set, SetQuadratic, Get, GetQuadratic

Selection of GMP functions

- GMP::Instance::Copy(GMP, name) -> GMP
- GMP::Instance::CreateDual(GMP, name) -> GMP
- GMP::Instance::CreateFeasibility(GMP, name, useMinMax) -> GMP
- GMP::Instance::CreatePresolved(GMP, name) -> GMP

Selection of GMP functions (2)

- GMP::Instance::SetOptionValue(GMP, OptionName, value)
- GMP::Solution::Check(GMP, solution, numInfeas, sumInfeas, skipObj)
- GMP::Instance::FixColumns(GMP1, GMP2, solution, variableSet, round)

Example: MIP Postsolve

- Why?
  - After solving MIP problem some integer variables might not have true integer values
- How?
  - Round and fix integer variables
  - Resolve resulting Relaxed MIP (RMIP)
- Can be used to calculate sensitivity information

MULTIPLE SOLUTIONS

Multiple solutions

- Solution repository of GMP can store multiple solutions
- Solution 1 is always the default/normal solution
- To investigate solutions you need to send them one by one to the variables
- Only one solution can be directly displayed at the same time in GUI

Displaying multiple solutions

```
GMP::Instance::Solve( myGMP );

NrSolutions := GMP::Solution::Count( myGMP );

cnt := 1;
while ( cnt <= NrSolutions ) do
    GMP::Solution::SendToModel( myGMP, cnt );

    display x.level;
    px(i,cnt) := x(i);

    cnt += 1;
endwhile;
```

x.solution(cnt).level;

Getting multiple solutions

- Incumbent callback MIP/MINLP (CPLEX, Gurobi, KNITRO)
- Solution pool (CPLEX)
- Give me *n* (best) solutions (BARON, CP Optimizer)
- Multi-start module (NLP)
- Use GMP::Instance::AddIntegerEliminationRows

SOLVER SESSIONS

Solver sessions

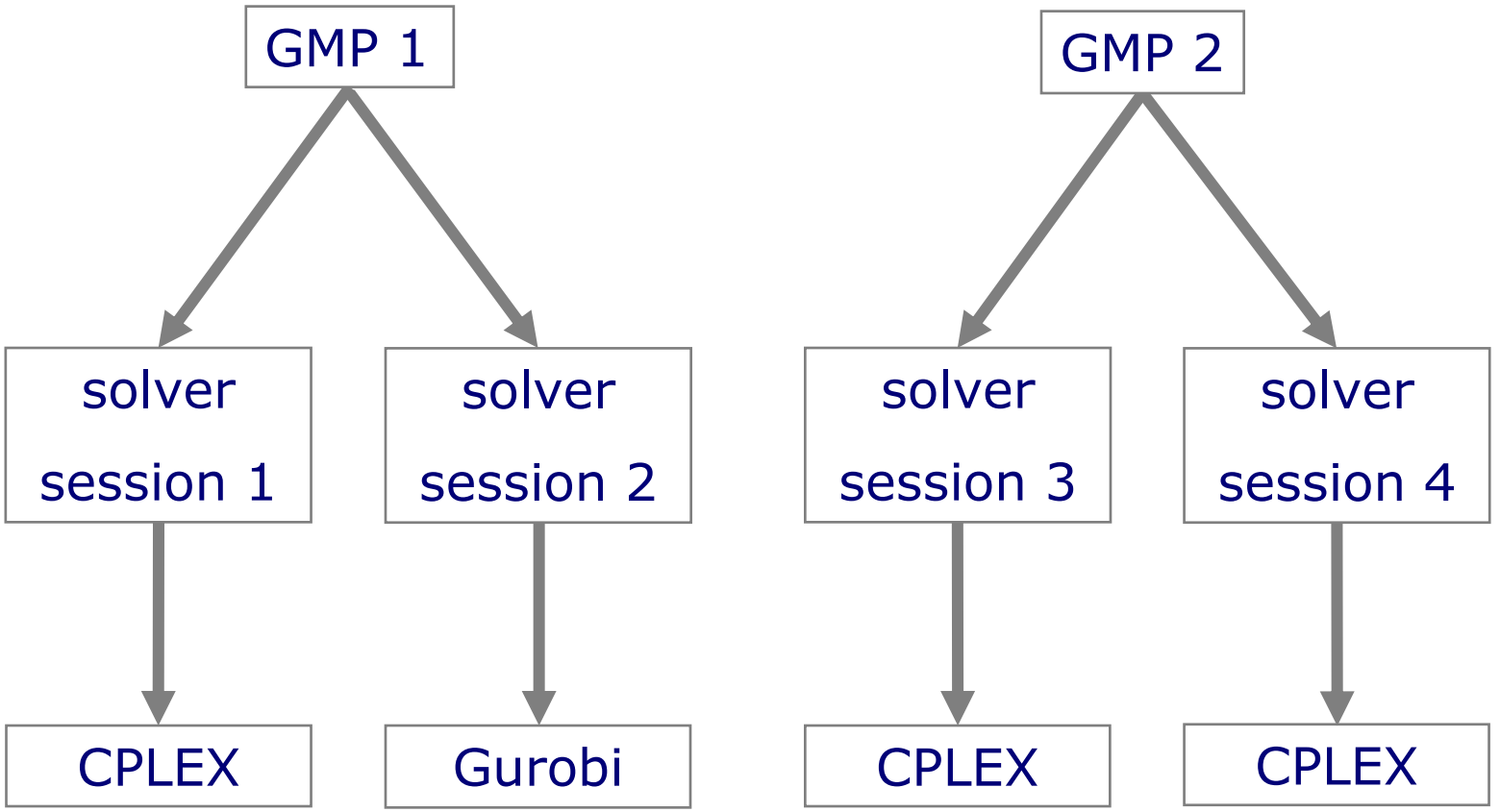
```
solve MathProgram;

    myGMP := GMP::Instance::Generate( MathProgram );
    GMP::Instance::Solve( myGMP );

        session := GMP::Instance::CreateSolverSession( myGMP );

        GMP::Solution::RetrieveFromModel( myGMP, 1 );
        GMP::Solution::SendToSolverSession( session, 1 );
        GMP::SolverSession::Execute( session );
        GMP::Solution::RetrieveFromSolverSession( session, 1 );
        GMP::Solution::SendToModel( myGMP, 1 );
```

Solver sessions



GMP cannot be modified if one of its solver sessions is running!

Example: re-solving large QP

- Solve large quadratic programming (QP) problem once a day
- Model data of QP changes slightly every day
- Can use starting solution, i.e., optimal basis of previous day
- But sometimes CPLEX then becomes very slow and it is faster to let CPLEX start from scratch

Asynchronous execute

```
session1 := GMP::Instance::CreateSolverSession( myGMP );
session2 := GMP::Instance::CreateSolverSession( myGMP );

GMP::SolverSession::SetOptionValue( session1, 'Advanced start', 1 );
GMP::SolverSession::SetOptionValue( session2, 'Advanced start', 0 );

GMP::SolverSession::AsynchronousExecute( session1 );
GMP::SolverSession::AsynchronousExecute( session2 );

css := GMP::SolverSession::WaitForSingleCompletion( AllSolverSessions );
if ( css = session1 ) then
   GMP::SolverSession::Interrupt( session2 );
else
   GMP::SolverSession::Interrupt( session1 );
endif;
```

Asynchronous solver possibilities

- No (technical) limitations:
  - CPLEX, Gurobi, XA, CONOPT, KNITRO
- Only one session:
  - CBC, IPOPT, SNOPT, MINOS, CP Optimizer
- Not possible:
  - BARON, PATH, AOA

References

- Language Reference
  - Implementing Advanced Algorithms for Mathematical Programs: Chapter 16 (AIMMS 4.1)
- Function Reference
  - The GMP library
- AIMMS Blog – tag GMP
- AIMMS Examples
  - http://www.aimms.com/downloads/application-examples

Announcement of next webinar

- The next webinar in this series, titled
  "Making the database connection"
  will be presented by Chris Kuip.

- Join us – Wednesday November 19, 2014:
  - 10 AM CET and
  - 8AM PDT/11AM EDT

- Register at
  http://www.aimms.com/news-events/webinars/