



Interactions between a Modeling System and Advanced Solvers

Jan-H. Jagla

jhjagla@gams.com

GAMS Software GmbH

www.gams.de

GAMS Development Corporation

www.gams.com

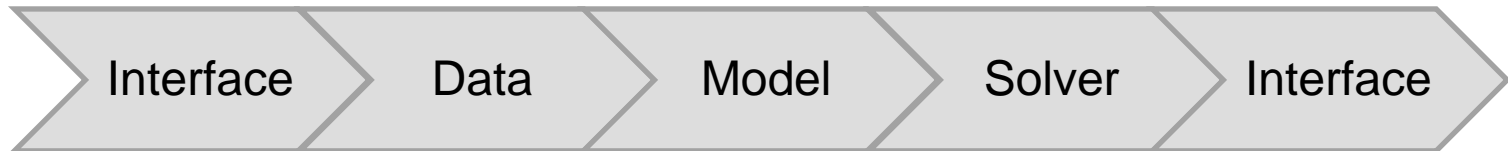




Agenda

GAMS' Fundamental concepts

- Different layers with separation of
 - model and data
 - model and solution methods
 - model and operating system
 - model and interface





Agenda

GAMS' Fundamental concepts

- Different layers with separation of
 - model and data
 - model and solution methods
 - model and operating system
 - model and interface





Solver Links

User

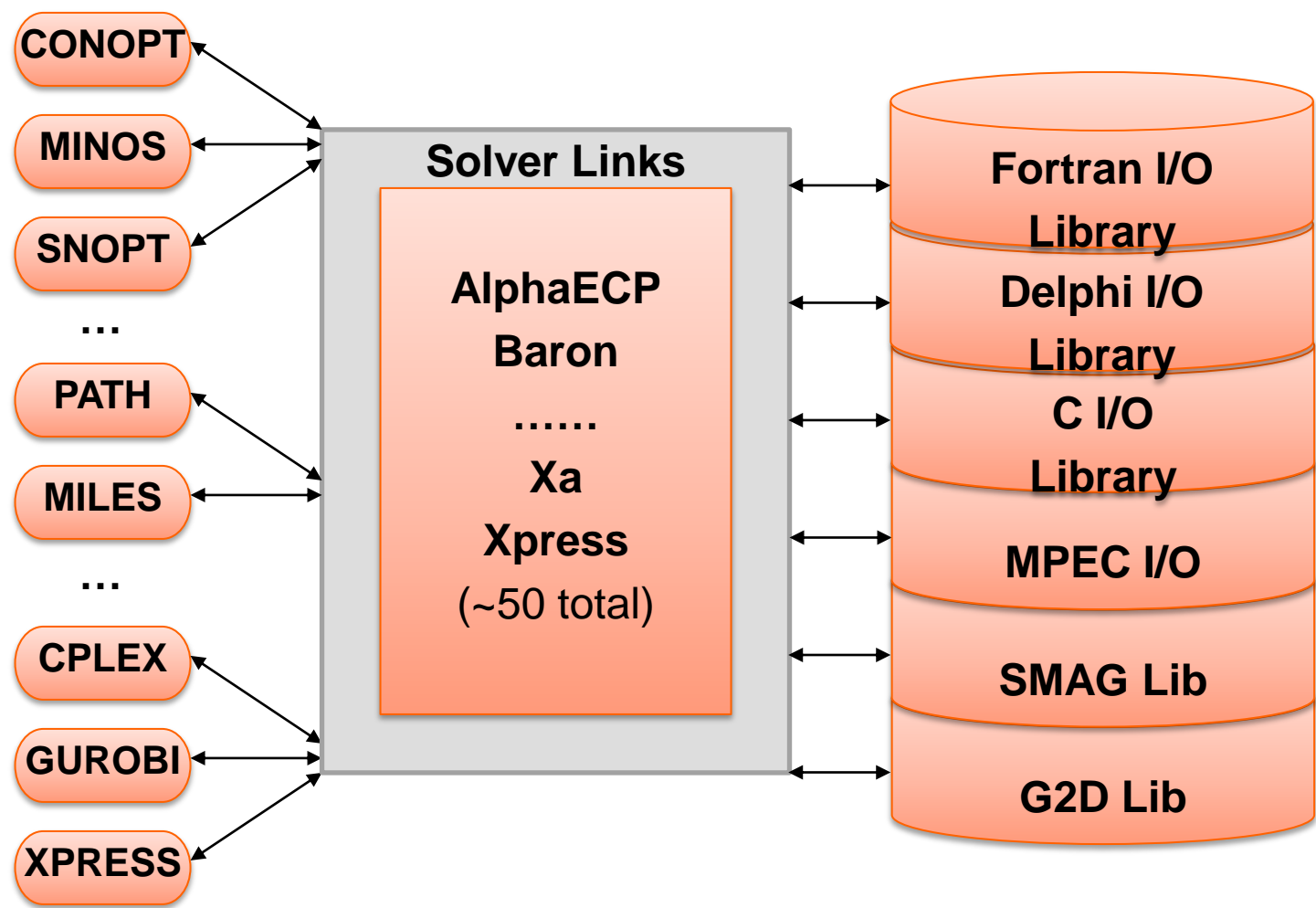
- Standardized solver interface allows “hassle free” replacement of solvers: *option nlp=conopt;*

Solver (Link) Developer

- IO Library provides access to
 - Matrix
 - Function/Gradient/Hessian evaluations
 - Solution file writer
 - Output handling
 - GAMS Options (e.g. resource limit)
 - ...



Reuse? What's that?!?

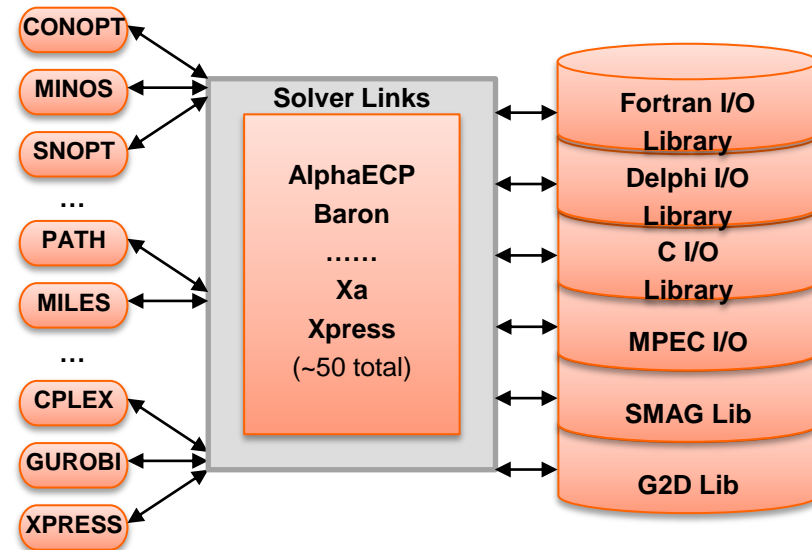




Multiple I/O Libraries

Advantages

- used for many years
- all architectures supported
- all GAMS-features available
- written by language experts
- ability to offer high quality link across platforms
→ has been one factor of success

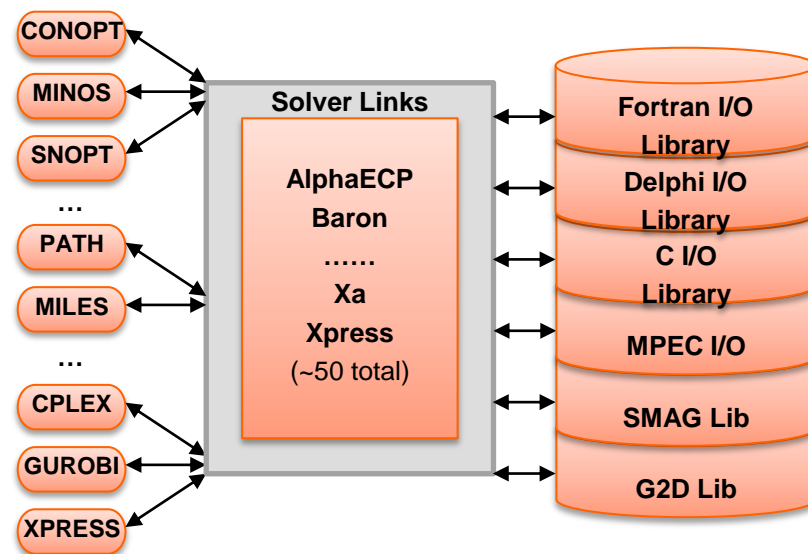




Multiple I/O Libraries

Disadvantages

- Not always intuitive to use Linking your Solver to GAMS - THE COMPLETE NOTES (160 pages !!)
- no automatic reformulation of objective func/var
- inconvenient to maintain
- painful to move 'inert mass' forward
- linking your solver (without buddy at GAMS) is very difficult





Single I/O Object

- Interfaces to all common programming languages C(++), C#, Delphi, Java, VB(A), Fortran, Python, ... **and** support on all platforms
- In-core communication between GAMS and the solver, making potentially large model scratch files unnecessary
- Make common link codes available inside object
- Support MP and Complementarity Models
- Automatic reformulations
- Open architecture



Gams Modeling Object



Single I/O Object

- Interfaces to all common programming languages C(++), C#, Delphi, Java, VB(A), Fortran, Python, ... **and** support on all platforms
- In-core communication between GAMS and the solver, making potentially large model scratch files unnecessary
- Make common link codes available inside object
- Support MP and Complementarity Models
- Automatic reformulations
- Open architecture



Gams Modeling Object



Automated Generation of APIs

'The GAMS Wrapper'

- API is defined using the GAMS language
- A tool written in GAMS is used to regenerate APIs for all languages
- Executed on request and nightly

```

* Properties reading/writing data that originally came from control vectors
gmoModelType .int. (r,w).getModelType ,w.modeltype )
gmoM .int. r .GetRows
gmoN .int. r .GetCols
gmoScaleOpt .int. (r,w).ScaleOpt
gmoSense .int. (r,direction ,w.setObjSense )
gmoObjVar .int. (r,w).GetObjVar ,w.SetObjVar )
gmoOptFile .int. (r,w).OptFile
gmoPriorOpt .int. (r,w).prioropt
gmoNLConst .int. (r,w).nlconst
gmoNZ .int. (r .GetNonZeros,w.NZ
gmoNLNZ .int. r .GetNLNonZeros
gmoNLM .int. r .GetNLRows
gmoNLN .int. r .GetNLCols
gmoObjRow .int. r .GetObjRow
gmoDictionary .int. (r,w).Dictionary
gmoHaveBasis .int. (r,w).havebasis

gmoNameOptFile .oSS. ( r .NameOptFile ,w.SetNameOptFile )
gmoNameSolFile .oSS. ( r .NameSolFile ,w.SetNameSolFile )
gmoNameLib .oSS. ( r .NameDll ,w.SetNameDll )
gmoNameMatFile .oSS. ( r .NameMatFile ,w.SetNameMatFile )
gmoNameDict .oSS. ( r .NameDict ,w.SetNameDict )
gmoNameParams .oSS. ( r .NameParams ,w.SetNameParams )
gmoNameInput .oSS. r .NameInput

Model Type
Number of equations
Number of variables
Scaling Flag
Direction of optimisation
Objective variable index
Optfile Number
Priority Flag
length of NL constant pool
Number of non zeros in constraints
Number of nonlinear non zeros in constral
Number of nonlinear rows
Number of nonlinear columns
Objective row index
Dictionary file written
Do we have basis

Option file name
Solution file name
External Function Library Name
Matrix file name
Dictionary file name
Params file name
Input file name

set if(en,tp,es,ta) function and procedures /
gmoLoadDataLegacy .(0,result.int,1,msg.oSS)
gmoInitData .(0,result.int,1,rows.int, 2,cols.int)
gmoCompleteData .(0,result.int,1,instname.CSS)
gmoQMaker .(0,result.int,1,density.D)
gmoSetObjQ .(0,result.int,1,colIdx.PLIA,2,rowIdx.PLIA,3,coef.PDA)
  
```

- A change in the definition of the API immediately makes it into all language interfaces
- No manual and therefore error-prone efforts required



Automated Generation of APIs

‘The GAMS Wrapper’

- Automated nightly testing
- API version checks
- Reusable for multiple GAMS component libraries
 - GMO
 - GAMS
 - GDX
 - Option

```

* Properties reading/writing data that originally came from control vectors
gmoModelType .int. ( r ,getModelType ,w.modeltype )
gmoM .int. r .GetRows
gmoN .int. r .GetCols
gmoScaleOpt .int. ( r,w ).ScaleOpt
gmoSense .int. ( r,direction ,w.setObjSense )
gmoObjVar .int. ( r ,GetObjVar ,w.SetObjVar )
gmoOptFile .int. ( r,w ).OptFile
gmoPriorOpt .int. ( r,w ).prioropt
gmoNLConst .int. ( r,w ).nlconst
gmoNZ .int. ( r ,GetNonZeros,w.NZ )
gmoNLNZ .int. r .GetNLNonZeros
gmoNLM .int. r .GetNLRows
gmoNLN .int. r .GetNLCols
gmoObjRow .int. r .GetObjRow
gmoDictionary .int. ( r,w ).Dictionary
gmoHaveBasis .int. ( r,w ).havebasis

gmoNameOptFile .OSS. ( r .NameOptFile ,w.SetNameOptFile )
gmoNameSolFile .OSS. ( r .NameSolFile ,w.SetNameSolFile )
gmoNameXLib .OSS. ( r .NameDll ,w.SetNameDLL )
gmoNameMatFile .OSS. ( r .NameMatFile ,w.SetNameMatFile )
gmoNameDict .OSS. ( r .NameDict ,w.SetNameDict )
gmoNameParams .OSS. ( r .NameParams ,w.SetNameParams )
gmoNameInput .OSS. r .NameInput

Model Type
Number of equations
Number of variables
Scaling Flag
Direction of optimisation
Objective variable index
Objective Number
Priority Flag
length of NL constant pool
Number of non zeros in constraints
Number of nonlinear non zeros in constral
Number of nonlinear rows
Number of nonlinear columns
Objective row index
Dictionary file written
Do we have basis

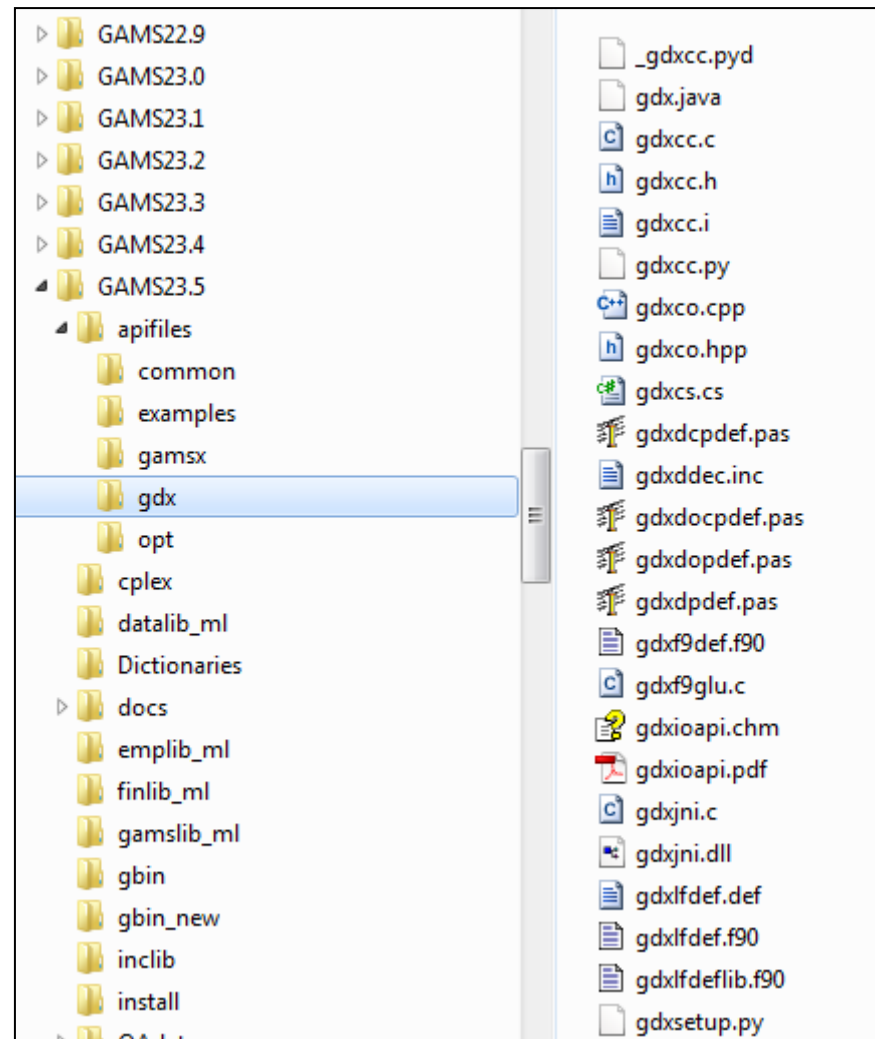
Option file name
Solution file name
External Function Library Name
Matrix file name
Dictionary file name
Params file name
Input file name

set if(en,tp,es,ta) function and procedures /
gmoLoadDataLegacy .(0,result.int,1,msg.oSS) Read GMO instance - Legacy Mode
gmoInitData .(0,result.int,1,rows.int, 2,cols.int) Initialises GMO data
gmoCompleteData .(0,result.int,1,instname.CSS) Complete GMO data instance needs lots of
gmoQMaker .(0,result.int,1,density.D) Create QP Info
gmoSetObjQ .(0,result.int,1,colIdx.PLIA,2,rowIdx.PLIA,3,coef.PDA) Get Q matrix for objective
    
```



Distributed GAMS APIs

- Component Libraries
 - GAMS
 - GDX
 - Option
- Supported languages
 - C, C++, C#
 - Delphi
 - Fortran
 - Java
 - VBA, VB.Net
 - Python
- Examples/Documentation





Single I/O Object

- Interfaces to all common programming languages C(++), C#, Delphi, Java, VB(A), Fortran, Python, ... **and** support on all platforms
- In-core communication between GAMS and the solver, making potentially large model scratch files unnecessary
- Make common link codes available inside object
- Support MP and Complementarity Models
- Automatic reformulations
- Open architecture



Gams Modeling Object



Solver Integration

```
solve mymodel minimizing z using lp  
mymodel.solveLink = {ChainScript, CallScript,  
    CallModule, AsyncGrid, AsyncSimulate, LoadLibrary};
```

- ChainScript: Solver process, GAMS vacates memory
 - + Maximum memory available to solver
 - + protection against solver failure (*hostile* link)
 - swap to disk
- Call{Script/Module}: Solver process, GAMS stays live
 - + protection against solver failure (*hostile* link)
 - + no swap of GAMS database
 - file based model communication



Solver Integration – cont.

- LoadLibrary: Solver DLL in GAMS process
 - + fast memory based model communication
 - + update of model object inside the solver (hot start)
 - not (yet) supported by all solvers
- trnsport.gms (LP) solved 500 times with CPLEX:

```
set ss /s1*s500/; loop(ss,  
    solve transport minimizing z using lp);
```

– ChainScript:	33.04 s (28.9s)*	
– CallModule:	13.78 s (12.7s)	
– LoadLibrary:	2.37 s (2.0s)	Cplex simplex time: 0.2 s
– Hot Start:	0.37 s (0.4s)	

* without Virus Scanner



Single I/O Object

- Interfaces to all common programming languages C(++), C#, Delphi, Java, VB(A), Fortran, Python, ... **and** support on all platforms
- In-core communication between GAMS and the solver, making potentially large model scratch files unnecessary
- Make common link codes available inside object
- Support MP and Complementarity Models
- Automatic reformulations
- Open architecture



Gams Modeling Object



In-house

GAMS Solver links

- New solver links are done with GMO
 - Couenne
 - Gurobi
 - OSI-based links to CPLEX, GUROBI, GLPK, MOSEK, XPRESS
 - ...
- Rewrite existing solver links using GMO
 - Coin (Bonmin, Cbc, Ipopt, OS)
 - Lindoglobal
 - Scip
 - ...



GAMS/Gurobi MIP Solver

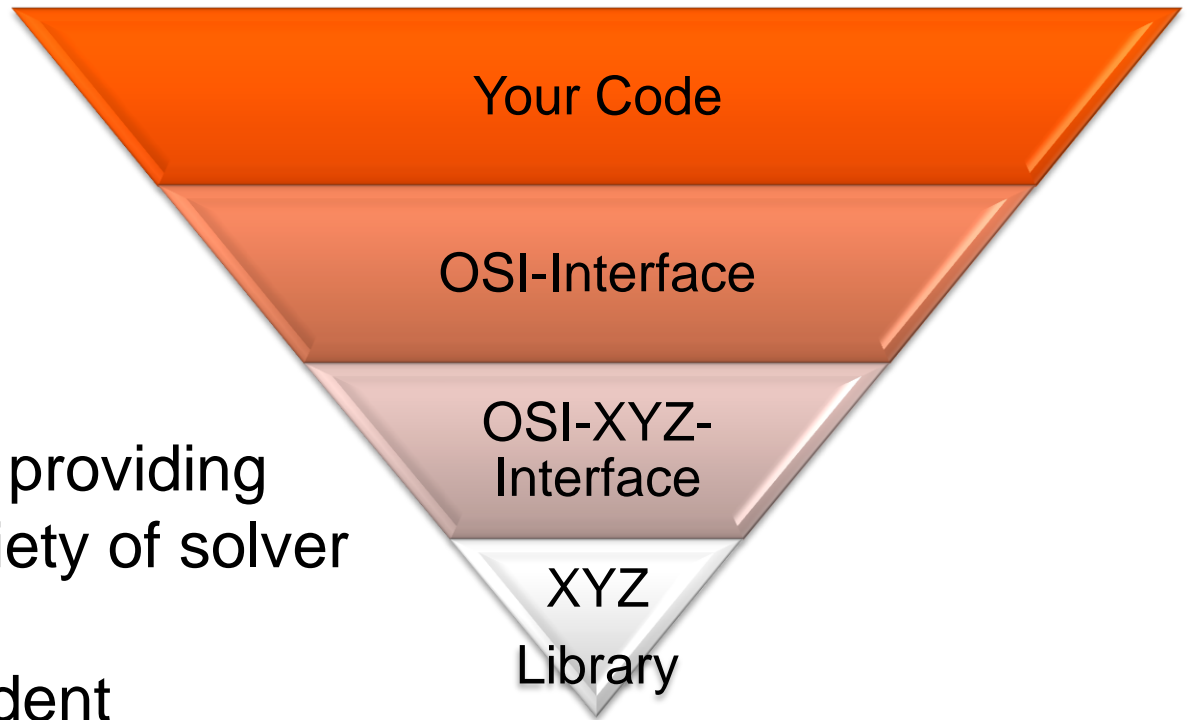


- A new MIP Solver with a Pedigree:
Zonghao **Gu**, Edward **Rothberg**, and Robert **Bixby**
former members of CPLEX' R&D team
- The Gurobi MIP solver
 - includes shared memory parallelism
 - is capable of simultaneously exploiting any number of cores
 - has a deterministic implementation
- GAMS/Gurobi link uses C Interface to GMO



Open Solver Interface-based links

- A standard API providing access to a variety of solver
- Solver independent



<http://projects.coin-or.org/Osi>

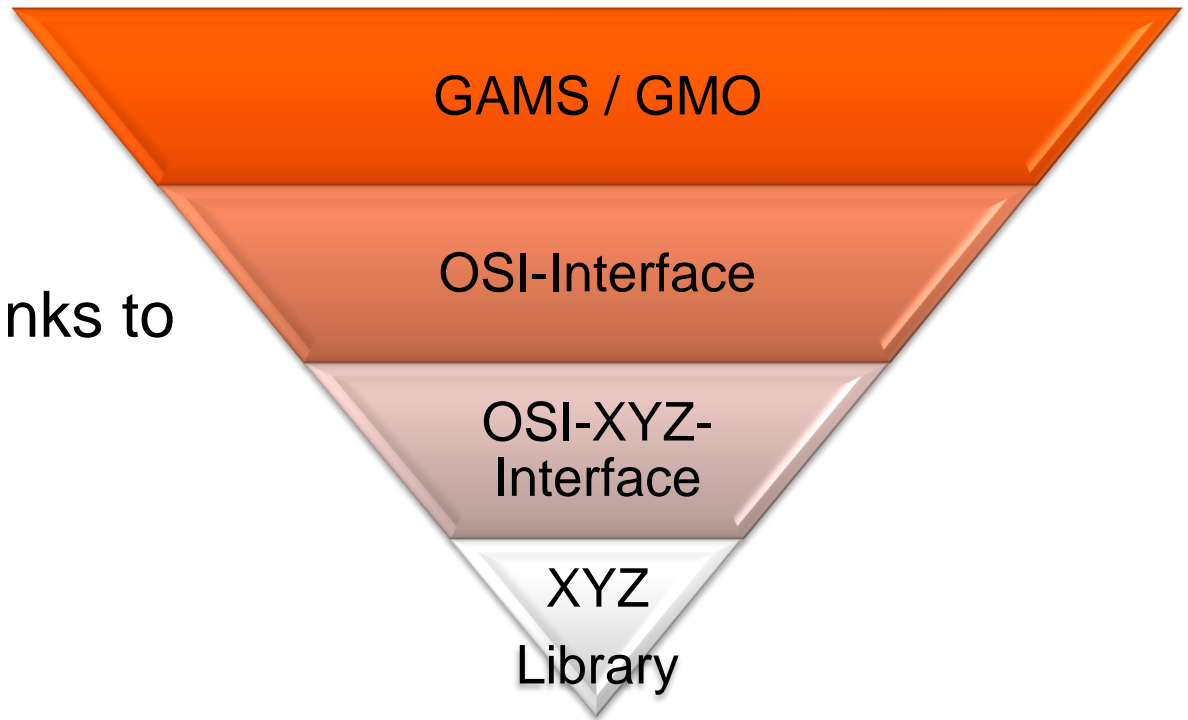


Open Solver Interface-based links

Free OSI-based links to

- CPLEX
- GUROBI
- MOSEK
- XPRESS

use C++ interface to GMO





Single I/O Object

- Interfaces to all common programming languages C(++), C#, Delphi, Java, VB(A), Fortran, Python, ... **and** support on all platforms
- In-core communication between GAMS and the solver, making potentially large model scratch files unnecessary
- Make common link codes available inside object
- Support MP and Complementarity Models
- Automatic reformulations
- Open architecture



Gams Modeling Object



External

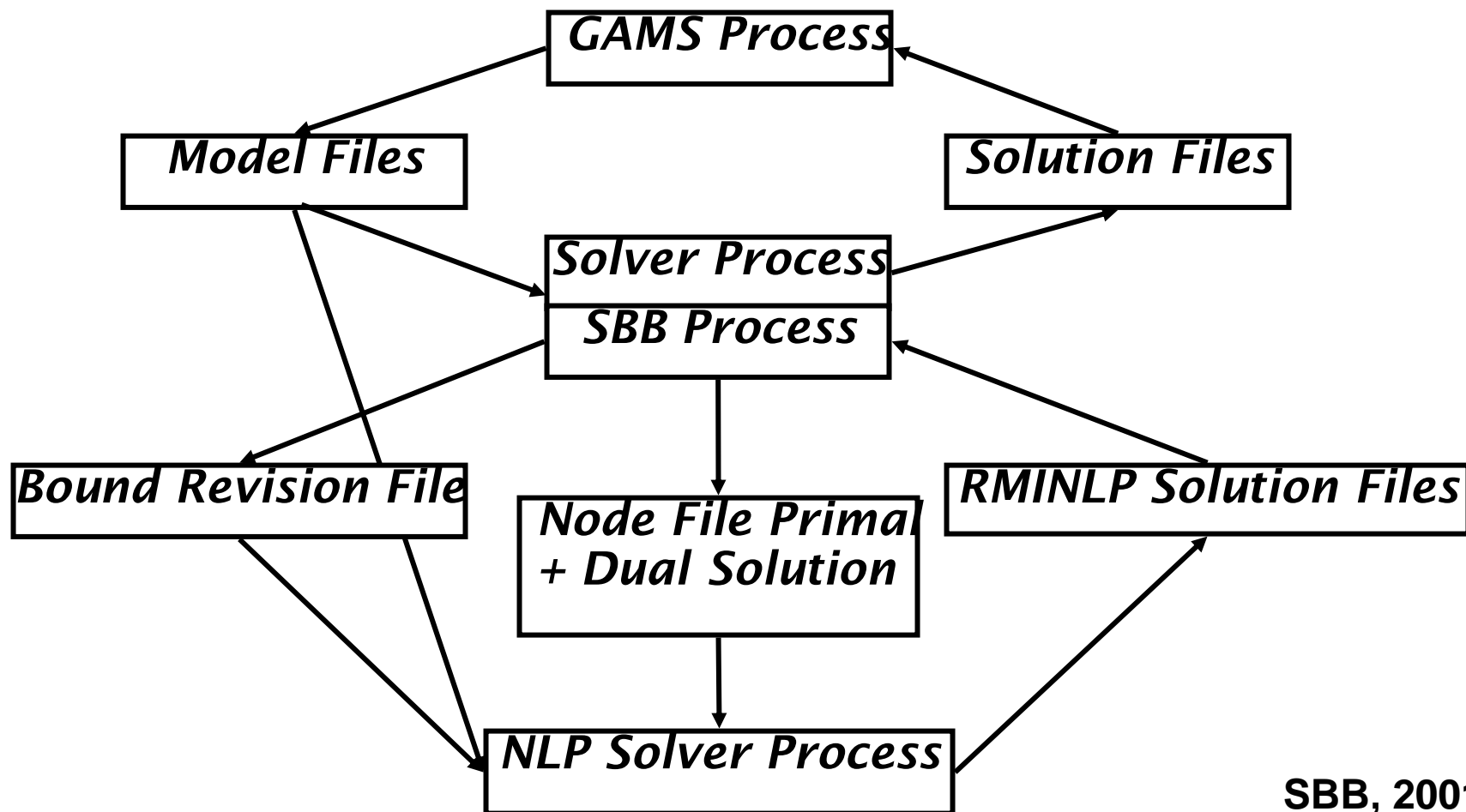
Open architecture

- Detach GMO from 'GAMS Environment'
- Ease linking of experimental solvers to GAMS
- Simplify to use GAMS as one piece of a puzzle

- E.g. sophisticated solvers use basic MP technology:
 - SBB (B&B requires NLP technology)
 - DICOPT (OA requires NLP+MIP)
 - BARON (requires NLP+LP)
 - LogMip (NLP+MIP)
 - DEA (LP)



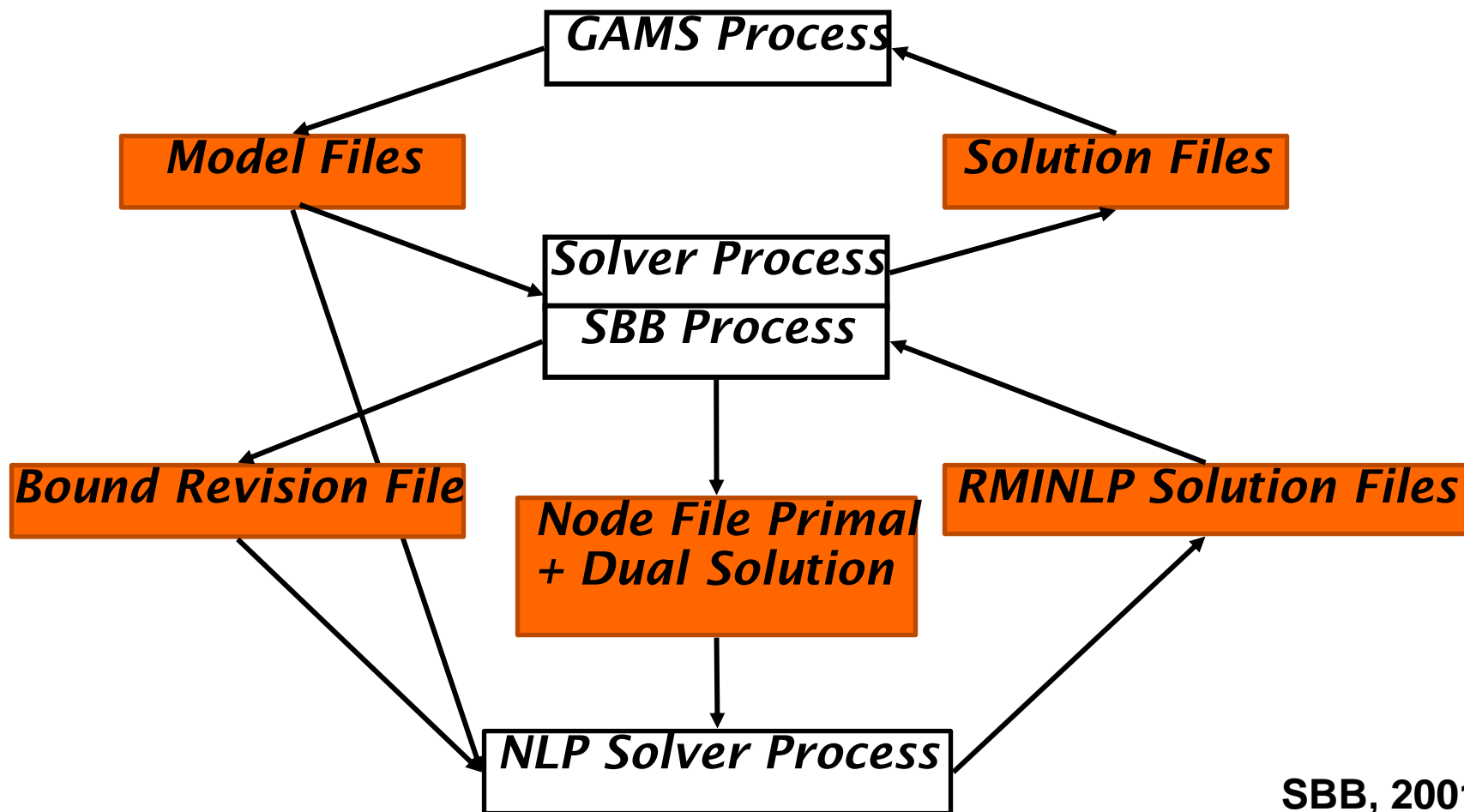
'Efficient' Implementation of B&B



SBB, 2001



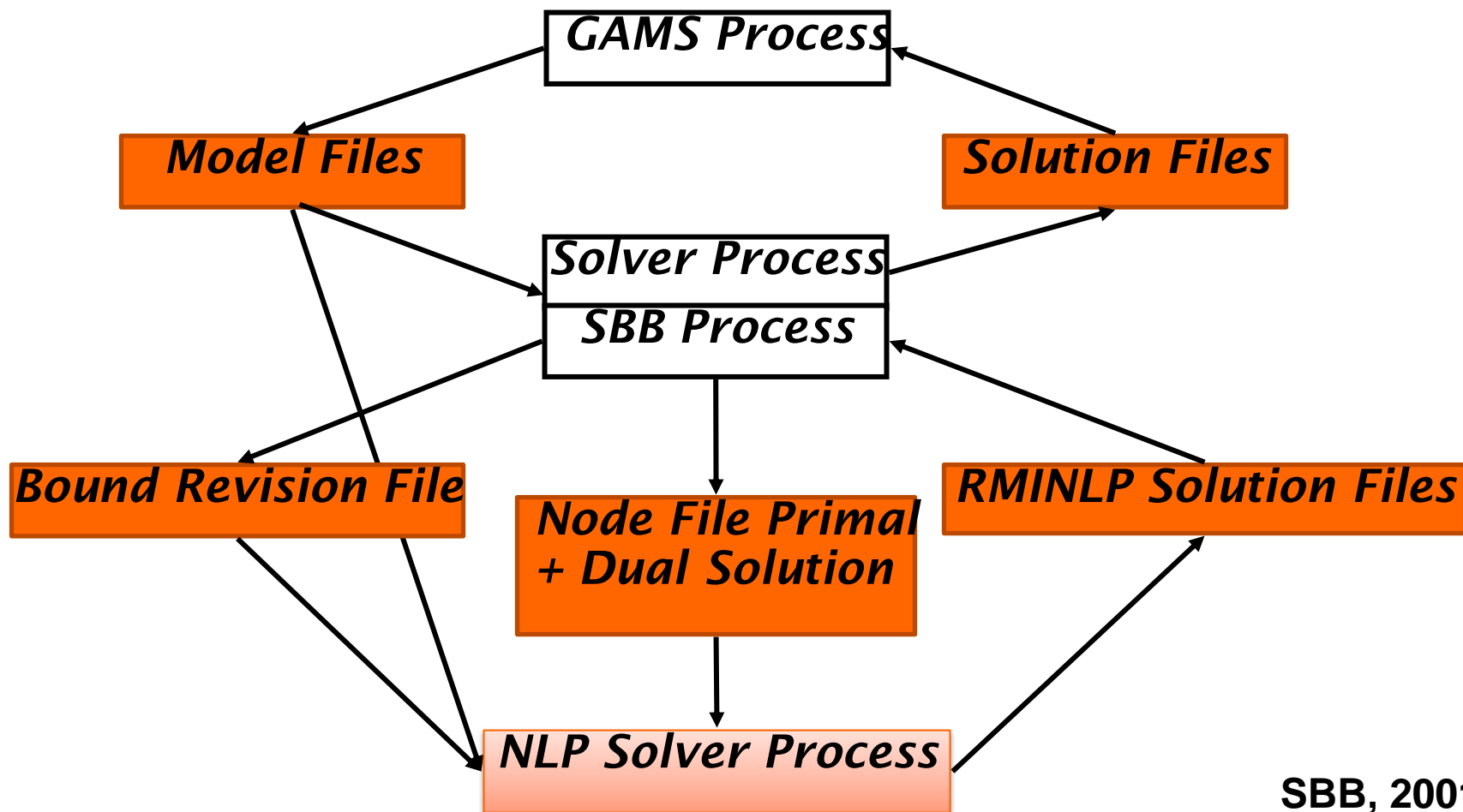
'Efficient' Implementation of B&B



SBB, 2001



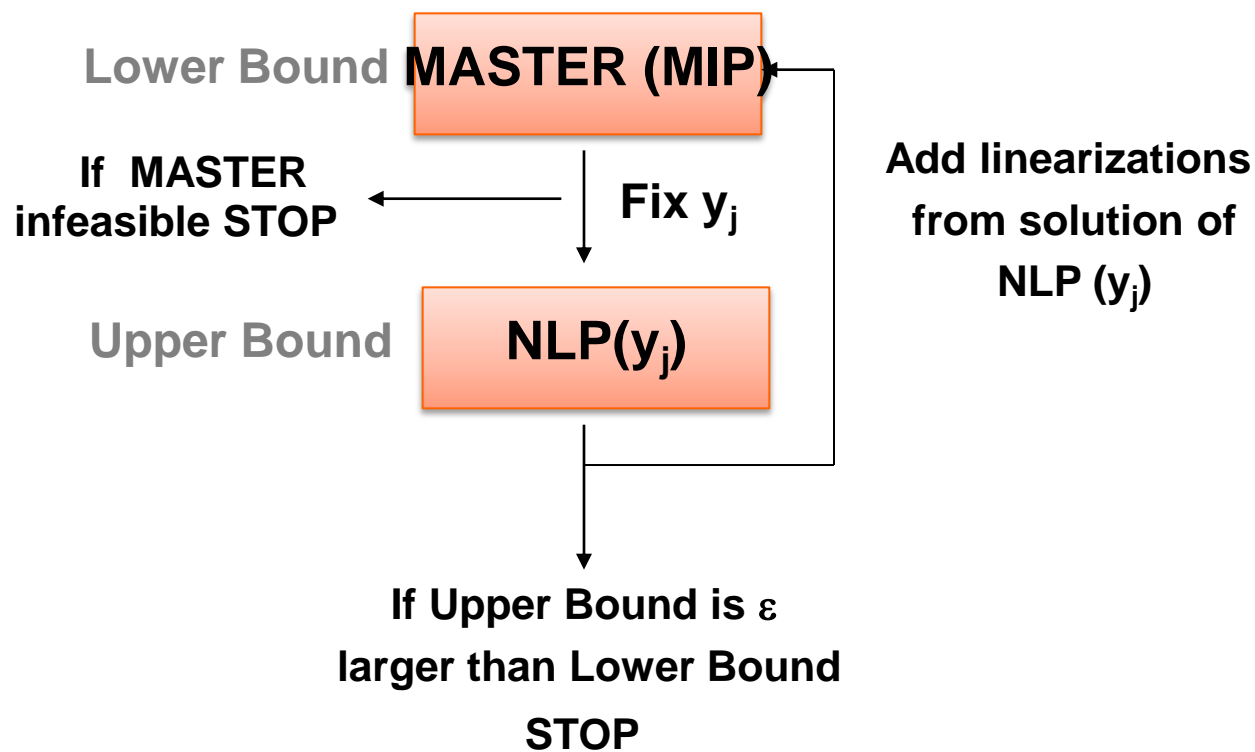
'Efficient' Implementation of B&B



SBB, 2001



Dicopt (Outer Approximation)





Series of NLP and MIP solves

```
--- DICOPT: Log File:
Major Major      Objective      CPU time      Itera-  Evaluation  Solver
Step  Iter        Function        (Sec)        tions      Errors
NLP   1           1.04923         0.02          38          0          conopt
MIP   1           9.07274         0.09          28          0          cplex
NLP   2           *Infeas*        0.00          10          0          conopt
MIP   2           13.02091        0.13          32          0          cplex
NLP   3           1.26864<       0.03          27          0          conopt
MIP   3           13.93760        0.11          29          0          cplex
NLP   4           *Infeas*        0.02           7          0          conopt
MIP   4           13.99258        0.11          19          0          cplex
NLP   5           *Infeas*        0.02          13          0          conopt
MIP   5           21.03812        0.11          23          0          cplex
NLP   6           1.26864         0.02          17          0          conopt
--- DICOPT: Terminating...
```

- Lots of file writing and reading to communicate between Dicopt, MIP, and NLP solver
- Basically start a whole new process over and over



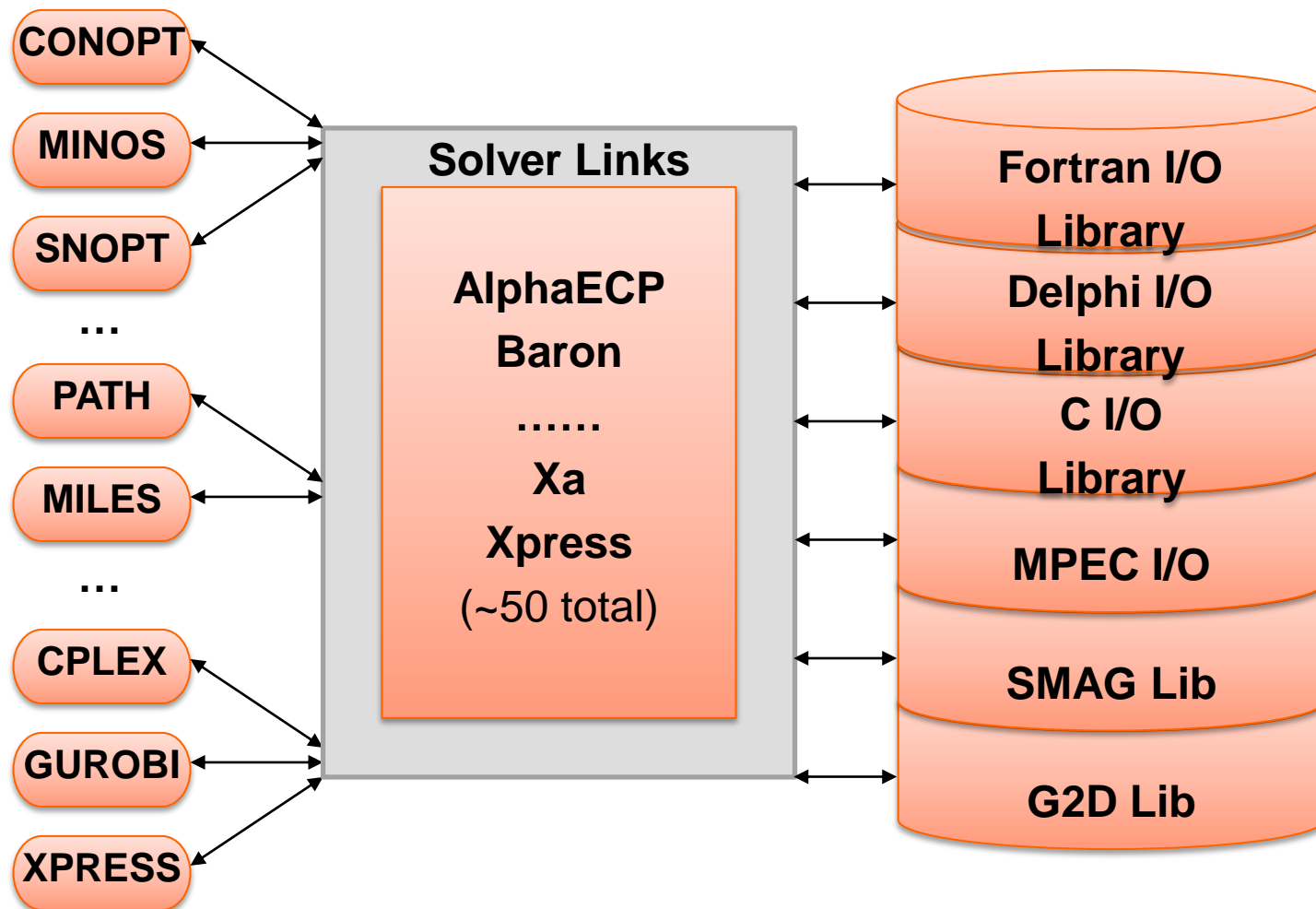
New Dicopt Implementation

Joined work with Ignacio Grossmann, Juan Pablo Ruiz (Carnegie Mellon University)

- Object Oriented
- Use C++ Interface to GMO
- Use standardized solver interface to call NLP/MIP solver in-core (pass GMO 'handle' on to solver)
- Algorithmic improvements

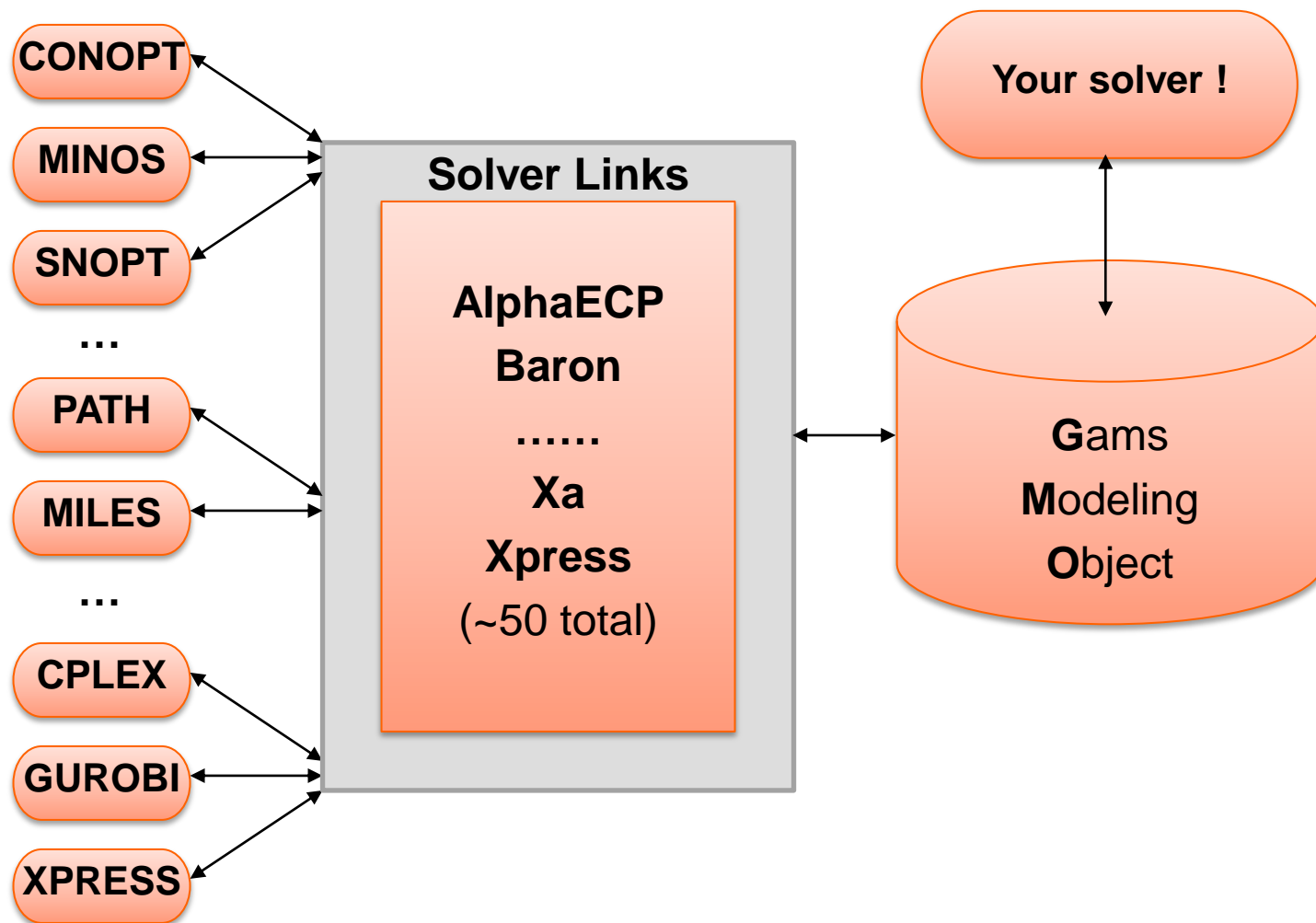


Summary





Summary





Summary

- GMO is part of GAMS distribution
- GMO is used by a variety of / will be used by all GAMS Solver Links
- GMO eases maintenance and makes development process more flexible
- GMO allows academics to quickly link their algorithm to GAMS
 - Use GAMS performance tools
 - Use GAMS QA tests
- GMO interfaces are not yet public but alpha version can be made available on request



Contacting GAMS

Europe

**GAMS Software GmbH
Eupener Str. 135-137
50933 Cologne
Germany**

Phone: +49 221 949 9170

Fax: +49 221 949 9171

<http://www.gams.de>

info@gams.de

USA

**GAMS Development Corp.
1217 Potomac Street, NW
Washington, DC 20007
USA**

Phone: +1 202 342 0180

Fax: +1 202 342 0181

<http://www.gams.com>

sales@gams.com

support@gams.com