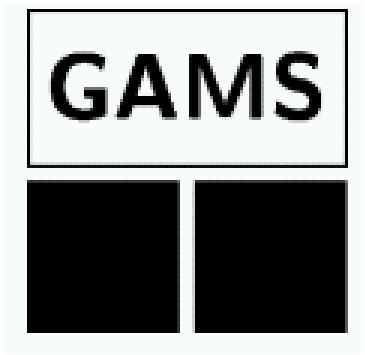


Bad Honnef, November 18th 2010

Jan-Hendrik Jagla jan@gams.com

Alexander Meeraus alex@gams.com

GAMS Development Corp.

GAMS Software GmbH www.gams.com

GAMS

Past Present Future

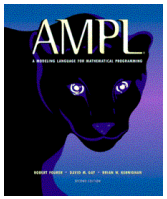
Algebraic Modeling System

What's that?

- Formulation mathematical optimization problems
- Notation similar to algebraic notation
- Ready-for-use links to state-of-the-art algorithms

➔Simplified model building
➔Efficient solution process

AIMMS THE MODELING SYSTEM 3



GAMS

. . .

General Algebraic Modeling System

- Roots: World Bank, 1976
- Went commercial in 1987
- GAMS Development Corporation (Washington, Houston)
- GAMS Software GmbH (Köln, Braunschweig)

- Broad academic & commercial user community and network

Model Structure

4 (c) Regional farmer employment accounting rows:

$$-RESr + 3 \sum_{d \in r} \sum_{q} dFLq + \sum_{d \in r} \sum_{t} dFLt = 0, \quad \text{each } r$$

$$-\left[\begin{array}{c}\text{Regional farmer}\\ \text{employment}\\ \text{activity}\end{array}\right] + 3\left[\begin{array}{c}\text{Sum over districts}\\ \text{and quarters of}\\ \text{quarterly farmer}\\ \text{employment}\end{array}\right]^{37}$$

$$+\left[\begin{array}{c}\text{Sum over districts}\\ \text{and months of}\\ \text{monthly farmer employment}\end{array}\right]^{37} = 0$$

1 (d) Total employment accounting row in man-years:

$$-12LMAN + \sum_{t} LMANt = 0$$

$$-12\left[\begin{array}{c}\text{Total employment}\\ \text{in man-years}\end{array}\right] + \left[\begin{array}{c}\text{Sum over months of}\\ \text{total employment}\\ \text{in man-months}\end{array}\right] = 0$$

12 (e) Total monthly employment accounting rows in man-months:

$$-2.2LMANt + \sum_{d} dDLt + \sum_{d} dFLq + \sum_{d} dFLt = 0,$$

each t and q such that $t \in q$

$$-2.2\left[\begin{array}{c}\text{Total}\\ \text{employment}\\ \text{in month } t\end{array}\right]^{38} + \left[\begin{array}{c}\text{Sum over districts of}\\ \text{day labor employment}\\ \text{in month } t\end{array}\right]$$

$$+\left[\begin{array}{c}\text{Sum over districts of}\\ \text{quarterly farmer}\\ \text{employment in the}\\ \text{quarter containing}\\ \text{month } t\end{array}\right] + \left[\begin{array}{c}\text{Sum over districts}\\ \text{of monthly farmer}\\ \text{employment}\end{array}\right] = 0$$

[37] In irrigation districts the quarterly contract device is used for farmers, but in non-irrigated districts farmers are assumed to be available on a monthly basis, so that seasonal migration to irrigated areas may occur.

[38] The activities for hiring farmers and day laborers are stated in units of tens of man-days per month (or quarter), and there are 22 working days per month; hence the conversion factor of 2.2 is required in the first term of this equation.

Model Data

Table 3
Sequence of standard operations for cotton cultivation (days of unskilled labor, machinery services, and draft animal services required per hectare by month)

Cultivation month and operation	Mechanized		Partially mechanized			Non-mechanized	
	Unskilled labor	Machinery	Unskilled labor	Machinery	Animals	Unskilled labor	Animals
1st Preparatory tasks		0.12		0.12		1.0	2.0
Fallow		0.5		0.5		3.0	6.0
Cross-plowing						2.5	5.0
Harrowing		0.2		0.2		0.5	1.0
Land levelling		0.25		0.25		1.0	2.0
Canal cleaning	1.0		1.0			1.0	
2nd Irrigation ditches	1.0	0.2	1.0	0.2		2.0	2.0
Forming borders [a]		0.2		0.2		2.0	
Linking borders [b]	1.0		1.0				
Water application	2.0		2.0			2.0	
Harrowing		0.2		0.2		2.0	4.0
Seeding and fertilization	0.2	0.2	0.2	0.2		4.0	
Maintenance of field works		0.2	0.2			2.0	
3rd Thinning plants	4.0		4.0			4.0	
Cultivation		0.2	2.0		4.0	2.0	4.0
Weeding	6.0		6.0			6.0	
Applications of insecticides (2) [c]							

Matrix Generator

```
Y(248)'X(248)
   IF (X(248),LT.0.5,AND,X(248),GT.,00) Y(248)'Z(248,1)*(1+X(248))
Y(249)'X(249)
   IF (X(249),LT.0.5,AND,X(249),GT.,00) Y(249)'Z(249,1)*(1+X(249))
Y(250)'X(250)
   IF (X(250),LT.0.5,AND,X(250),GT.,00) Y(250)'Z(250,1)*(1+X(250))
Y(251)'X(251)
   IF (X(251),LT.0.5,AND,X(251),GT.,00) Y(251)'Z(251,1)*(1+X(251))
Y(252)'X(252)
   IF (X(252),LT.0.5,AND,X(252),GT.,00) Y(252)'Z(252,1)*(1+X(252))
Y(253)'X(253)
   IF (X(253),LT.0.5,AND,X(253),GT.,00) Y(253)'Z(253,1)*(1+X(253))
Y(254)'X(254)
   IF (X(254),LT.0.5,AND,X(254),GT.,00) Y(254)'Z(254,1)*(1+X(254))
Y(255)'Y(266)+Y(267)
Y(256)'X(256)
   IF (X(256),LT.0.5,AND,X(256),GT.,00) Y(256)'Z(256,1)*(1+X(256))
Y(257)'X(257)
   IF (X(257),LT.0.5,AND,X(257),GT.,00) Y(257)'Z(257,1)*(1+X(257))
Y(258)'X(258)
   IF (X(258),LT.0.5,AND,X(258),GT.,00) Y(258)'Z(258,1)*(1+X(258))
Y(259)'X(259)
   IF (X(259),LT.0.5,AND,X(259),GT.,00) Y(259)'Z(259,1)*(1+X(259))
Y(260)'X(260)
   IF (X(260),LT.0.5,AND,X(260),GT.,00) Y(260)'Z(260,1)*(1+X(260))
Y(261)'Y(63)
Y(262)'X(262)
   IF (X(262),LT.0.5,AND,X(262),GT.,00) Y(262)'Z(262,1)*(1+X(262))
Y(263)'X(263)
   IF (X(263),LT.0.5,AND,X(263),GT.,00) Y(263)'Z(263,1)*(1+X(263))
Y(264)'X(264)
   IF (X(264),LT.0.5,AND,X(264),GT.,00) Y(264)'Z(264,1)*(1+X(264))
Y(265)'X(265)
   IF (X(265),LT.0.5,AND,X(265),GT.,00) Y(265)'Z(265,1)*(1+X(265))
Y(266)'X(266)
   IF (X(266),LT.0.5,AND,X(266),GT.,00) Y(266)'Z(266,1)*(1+X(266))
Y(267)'X(267)
```

```
CUMPPB
CUMPIB
CUMNEI
CUMDME
CUMTRO
CUMTHR
CUMFCU
CUMFIV
CUMLCT
CUMLCG
CUMDLS
CU b=
C= -
EXPORT
NETDII
NETDFI
MRKRMT
NETTRN
OFFCUR
OFFCAP
```

Matrix Generator Input

```
   25     1     8     0     0     0     1      AGGREGAT
      0.12                    0.0165
ALA ALG ALV ARO AZU CAR CEG CHV FRI GAR JIT JON MAI MAT MEL P
PLU SAL SAN SOR SOT SOY TRI
    0.0286
     99999
AZU AZU    -0.25       1.0         0.0070     2627020.
JIT JIT    -0.4        1.0         0.1150     174752.
PEP PEP    -0.6        1.0         0.0590     19.
PLU PLU    -1800.      1.0         0.5770     85209.
CCC
CHI        1     -0.2
CHV         0.1500     14.459      1.0
FOR        6     -0.3
SOR         0.0630     285.818     1.0
CEG         0.0930     0.665       1.0
ALV         0.0100     226.109     1.0
ALA         0.0400     179.019     1.0
GAR         0.0990     1.427       1.0
MAI         0.0860     77.997      1.0
FEC        4     -0.3
FRI         0.1830     33.001      1.0
ARO         0.1220     126.197     1.0
PAP         0.0930     27.138      1.0
GAR         0.0990     0.158       1.0
GRA        2     -0.1
MAI         0.0860     132.804     1.0
TRI         0.0800     343.979     1.0
FRU        2     -2.0
SAN         0.0780     10.850      1.0
MEL         0.0680     6.9350       1.0
OLE        4     -1.2
SAL         0.0830     193.910     1.0
JON         0.2410     9.224       1.0
CAR         0.1550     73.490      1.0
SOY         0.1600     57.220      1.0
END
ALA         .02        0.0
ALV          .005       .06
```

MPS File – Column Section

```
X,ASGHC2    B,AS,,C2        -1,00000
X,ASGHC2    A,TRA            6,98400
X,ASGHC3    D,,,GH,N         0,33500
X,ASGHC3    R,,,GHC3         1,00000
X,ASGHC3    B,AS,,C3        -1,00000
X,ASGHC3    A,TRA            6,98400
X,ASGHAS    D,,,GH,N         0,20600
X,ASGHAS    R,,,GHAS         1,00000
X,ASGHAS    B,AS,,AS        -1,00000
X,ASGHAS    A,TRA            6,98400
X,ASGHS1    D,,,GH,P         0,15000
X,ASGHS1    R,,,GHS1         1,00000
X,ASGHS1    B,AS,,S1        -1,00000
X,ASGHS1    A,TRA            6,98400
X,ASGHCN    R,,,GHCN         1,00000
X,ASGHCN    B,AS,,CN        -1,00000
X,ASGHCN    A,TRA            6,98400
X,ASKSC1    D,,,KS,N         0,26000
X,ASKSC1    R,,,KSC1         1,00000
X,ASKSC1    B,AS,,C1        -1,00000
X,ASKSC1    A,TRA            7,56000
X,ASKSC2    D,,,KS,N         0,31000
X,ASKSC2    R,,,KSC2         1,00000
X,ASKSC2    B,AS,,C2        -1,00000
X,ASKSC2    A,TRA            7,56000
X,ASKSC3    D,,,KS,N         0,33500
X,ASKSC3    R,,,KSC3         1,00000
X,ASKSC3    B,AS,,C3        -1,00000
X,ASKSC3    A,TRA            7,56000
X,ASKSAS    D,,,KS,N         0,20600
X,ASKSAS    R,,,KSAS         1,00000
X,ASKSAS    B,AS,,AS        -1,00000
X,ASKSAS    A,TRA            7,56000
X,ASKSS1    D,,,KS,P         0,15000
X,ASKSS1    R,,,KSS1         1,00000
X,ASKSS1    B,AS,,S1        -1,00000
X,ASKSS1    A,TRA            7,56000
X,ASKSCN    R,,,KSCN         1,00000
X,ASKSCN    B,AS,,CN        -1,00000
```

MPS Revision File

```
     BRANCH     *          MAJERR
     NEXT
     REVISE     REV5       TAPE14


                                        ***** CARD READ SUMMARY ****
HEADER, CARD NO=     1    $NAME          REVA
HEADER, CARD NO=     2    $COLUMNS
HEADER, CARD NO=     3    $  MODIFY
HEADER, CARD NO=     6    $RHS
HEADER, CARD NO=     7    $  MODIFY
HEADER, CARD NO=    18    $ENDATA
HEADER, CARD NO=    19    $NAME          REV1
HEADER, CARD NO=    20    $COLUMNS
HEADER, CARD NO=    21    $  MODIFY
HEADER, CARD NO=    42    $ENDATA
HEADER, CARD NO=    43    $NAME          REV2
HEADER, CARD NO=    44    $COLUMNS
HEADER, CARD NO=    45    $  MODIFY
HEADER, CARD NO=    51    $ENDATA
HEADER, CARD NO=    52    $NAME          REV4
HEADER, CARD NO=    53    $RHS
HEADER, CARD NO=    54    $  MODIFY
HEADER, CARD NO=    68    $ENDATA
HEADER, CARD NO=    69    $NAME          REV5
HEADER, CARD NO=    70    $RHS
HEADER, CARD NO=    71    $  MODIFY
        CARD NO=    72    $    RHS1      CLA,V,01      5,03328
        CARD NO=    73    $    RHS1      CLA,V,02      5,03328
        CARD NO=    74    $    RHS1      CLA,V,03      5,03328
        CARD NO=    75    $    RHS1      CLA,V,04      5,03328
        CARD NO=    76    $    RHS1      CLA,V,05      5,03328
        CARD NO=    77    $    RHS1      CLA,V,06      5,03328
        CARD NO=    78    $    RHS1      CLA,V,07      5,03328
        CARD NO=    79    $    RHS1      CLA,V,08      5,03328
        CARD NO=    80    $    RHS1      CLA,V,09      5,03328
        CARD NO=    81    $    RHS1      CLA,V,10      5,03328
        CARD NO=    82    $    RHS1      CLA,V,11      5,03328
        CARD NO=    83    $    RHS1      CLA,V,12      5,03328
        CARD NO=    84    $    RHS1      CLA,V,70     60,39936
HEADER, CARD NO=    85    $ENDATA
```

MPS Output

```
DATE 07/30/76  TIME 22.12.21                                   C O L U M N S    APEX-III  1.000  PAGE

   PRINT OPTION = COMPLETE OUTPUT W/SPECIAL                                     TIVE =              28.18489
NAME = CENTRAL      OBJ  = OBJ          RHS  = RHS1                             1.0000  RPSRHS  =     1.0000
DIR  = MAXIMIZE     COBJ =              CRHS =            BND = LIMITS          0.0000  RPCHRMS =     0.0000
                                                          RNG =
```

NUMBER	NAME	TYPE	STATUS	COL ACTIVITY	OBJ COEF	D UPPER	MARGINAL
101	CBE1V..	PL	LOWER	.	-47.80000	+INF	-6.46851
102	CBE2F..	PL	ACTIVE	.00087	-701.00000	+INF	.
103	CBE3C..	PL	ACTIVE	.	-10330.60000	+INF	.
104	CBE4F..	PL	LOWER	.	-2429.70000	+INF	-912.25118
105	CBE5C..	PL	LOWER	.	-9418.00000	+INF	-2342.38642
106	CBE6C..	PL	ACTIVE	.	-5118.00000	+INF	.
107	CBE7C..	PL	ACTIVE	.06067	-13.20000	+INF	.
108	CSG.V..	PL	ACTIVE	.	-231.57000	+INF	.
109	CSG.F..	PL	ACTIVE	.00226	-231.57000	+INF	.
110	CPO.V..	PL	ACTIVE	.	-139.67000	+INF	.
111	CPO.F..	PL	ACTIVE	.00002	-139.67000	+INF	.
112	CPO.C..	PL	ACTIVE	.00045	-139.67000	+INF	.
113	CEG.V..	PL	ACTIVE	.	-76.71000	+INF	.
114	CEG.F..	PL	ACTIVE	.00025	-76.71000	+INF	.
115	CEG.C..	PL	ACTIVE	.00128	-76.71000	+INF	.
116	CDA.CX.	PL	ACTIVE	.07685	12.91000	+INF	.
117	COF.CX.	PL	LOWER	.	180.74000	+INF	-87.19134
118	COC.CX.	PL	LOWER	.	167.83000	+INF	-256.39963
119	COS.CX.	PL	ACTIVE	.06968	121.35000	+INF	.
120	COL.CX.	PL	ACTIVE	.00225	91.66000	+INF	.
121	CWS.CX.	PL	ACTIVE	.00606	109.74000	+INF	.
122	CWL.CX.	PL	ACTIVE	.00748	77.46000	+INF	.

WB Old Slide 1

PLANNING PROBLEM AND OBJECTIVES INITIALLY OFTEN

UNSTRUCTURED

ILL-DEFINED

CONFLICTING

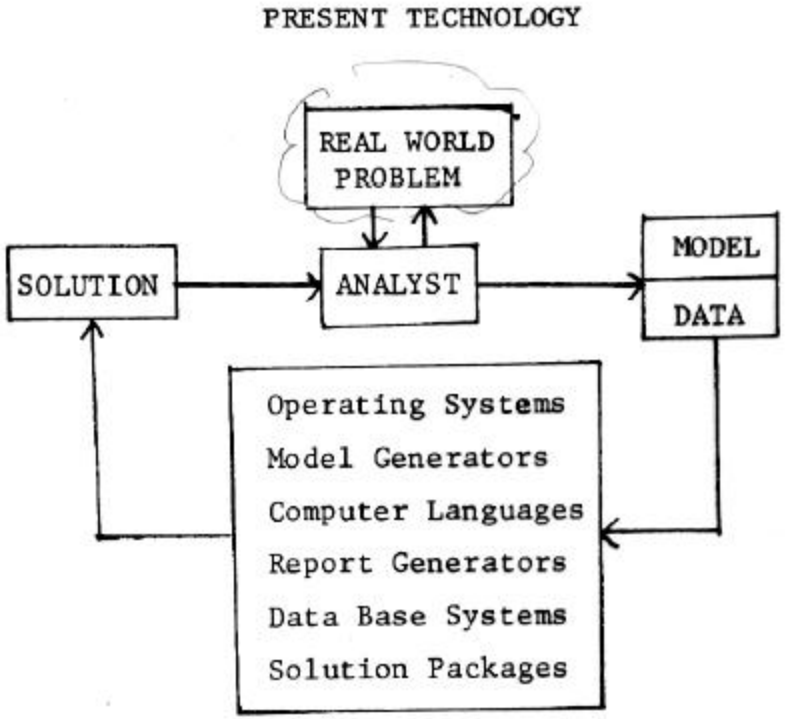
UNCERTAIN

CHANGING

EMOTIONAL

MATHEMATICAL MODEL USED TO RECOGNIZE AND FORMULATE PROBLEMS, DEFINE ISSUES AND EXPLORE SOLUTION SPACE

WB Old Slide 2



RESULT: - Drain of resources (technical, time, money)

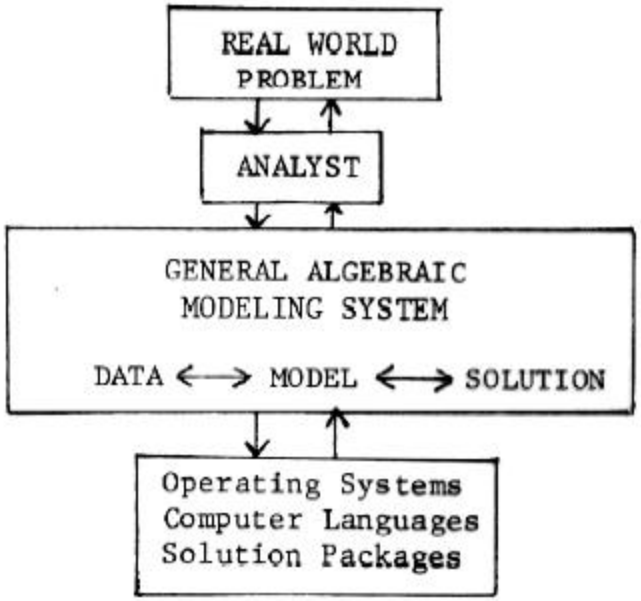
- Essentially no documentation

WB Old Slide 3

MAJOR CONSTRAINTS :

- COST
- SKILLS
- TIME
- TOOLS
- DOCUMENTATION
- TRUST
- .
- .
- .

WB Old Slide 4



RESULT:
- Limited drain of resources
- Same representation of models for humans and machines
- Model representation is also model documentation

WB Old Slide 5

DEVELOPMENT OF GAMS

Phase 1 (1978)

- The system can be used to represent and analyze any algebraic model (be it linear or nonlinear)
- The system can perform algebraic manipulations on all data
- The system can generate and solve linear programs automatically
- The system can generate reports on data and solutions via simple 'display' statements

WB Old Slide 6

DEVELOPMENT OF GAMS

Phase 2 (1979)

- The system can generate and solve nonlinear programs
- The system will provide links to special-purpose algorithms for econometric problems, network problems, etc.
- Appropriate extensions to the language will be made as the need arises

WB Old Slide 7

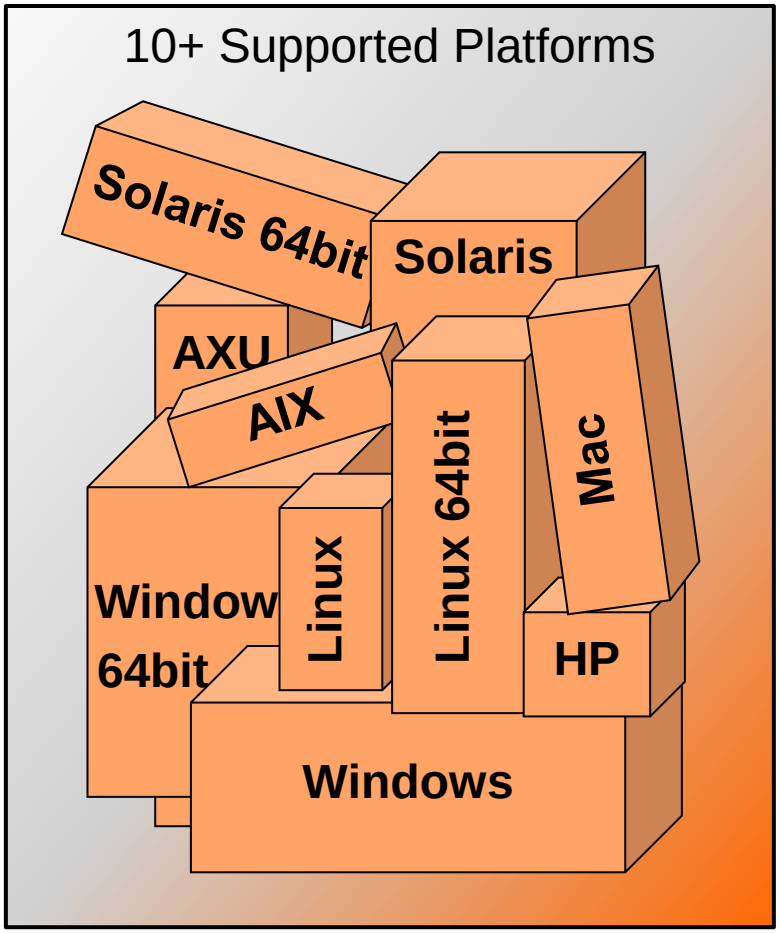
DEVELOPMENT OF GAMS

Phase 3 (?)

- Automatic structure recognition
- Internal generation of *exact* point-derivatives
- Improved data-base design with e.g. unit analysis, and links to existing data bases
- Availability of GAMS on different machines
- World-wide availability of the system so that it can be used as a market for testing models and algorithms

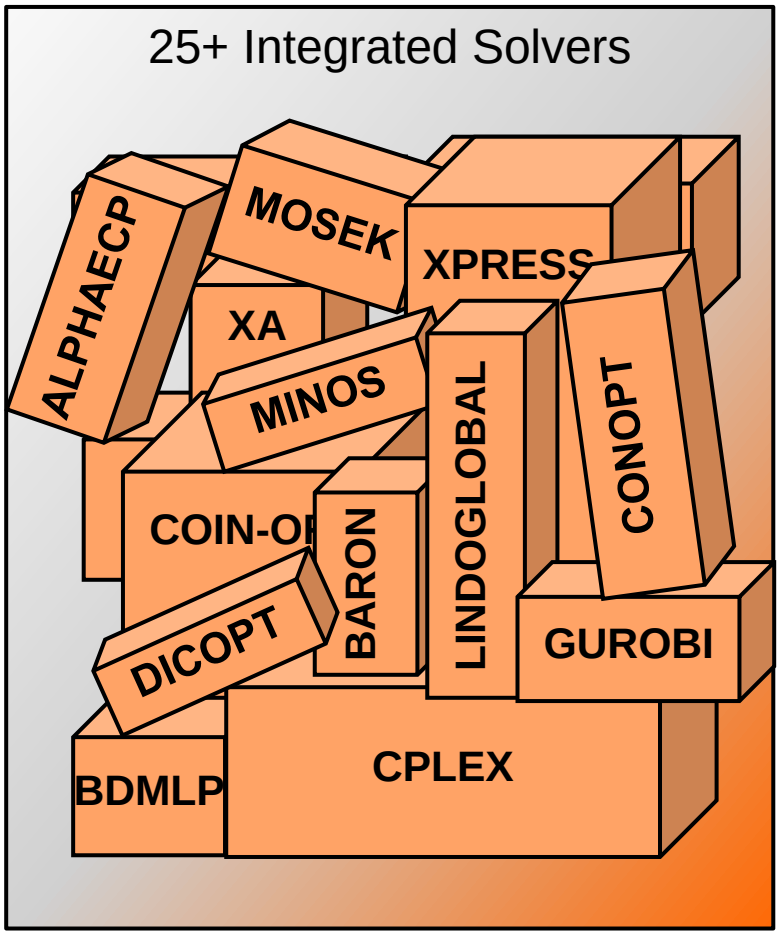
GAMS' Fundamental concepts

- **Platform independence**
- Hassle-free switch of solution methods
- Open architecture and interfaces to other systems
- Balanced mix of declarative and procedural elements



GAMS' Fundamental concepts

- Platform independence
- **Hassle-free switch of solution methods**
- Open architecture and interfaces to other systems
- Balanced mix of declarative and procedural elements



GAMS' Fundamental concepts

- Platform independence
- Hassle-free switch of solution methods
- **Open architecture and interfaces to other systems**
- Balanced mix of declarative and procedural elements

Gams Data eXchange (GDX)

API's

- Gams Data eXchange (GDX)
- Options
- Gams Modeling Object (GMO)
- Gams Environment (GEV)
- …

Layers of separation

GAMS' Fundamental concepts

- **Layers with separation of**
  - model and data
  - model and solution methods
  - model and operating system
  - model and interface

→ **Models benefit from**
  - advancing hardware
  - enhanced / new solver technology
  - improved / upcoming interfaces to other systems

GAMS' Fundamental concepts

- Platform independence
- Hassle-free switch of solution methods
- Open architecture and interfaces to other systems
- **Balanced mix of declarative and procedural elements**

Declarative Elements

- Sets
- Parameters
- Variables
- Equations
- Models
- …

Procedural Elements

- loops
- if-then-else
- …

Problem 1

- Data transfer between different systems slow, error prone and bulky.
- Application (real time) require the capture of data instances that can be analyzed off-line in other environments.
- Management of name space mappings between different problems and their transformations into other data representations.
- Separate the model from its environment
- Search for a common low level high performance data container

Problem 2

- Building and maintaining solver specific links in different programming languages became a huge resource sink and made the introduction of new features difficult.

  → Simplify the building and maintaining of solver links
  → Manage multiple interacting models
  → Minimize the solver specific tailoring
  → Maintain one source only
  → Wrap automatically for different languages.

- Share libraries between the data management part of a modeling system and the solver. Example: function evaluations, first and second order derivatives, intervals,..

- Ease linking of experimental (meta-)solvers to GAMS

Problem 3

- Problems may contain
  - Complementarity
  - Hierarchy
  - Interacting agents
  - Risk measures
  - Logic relationships
- Cannot be expressed with current modeling languages and have no direct solution method.
- Example: General equilibrium models are a transformation from multi agent optimization/variational problems into a single mixed complementarity model.
- How to automate the transformations by annotations of existing optimization models that convey model structure to the solver.


GAMS

GAMS

Features you might not know

Problem 1

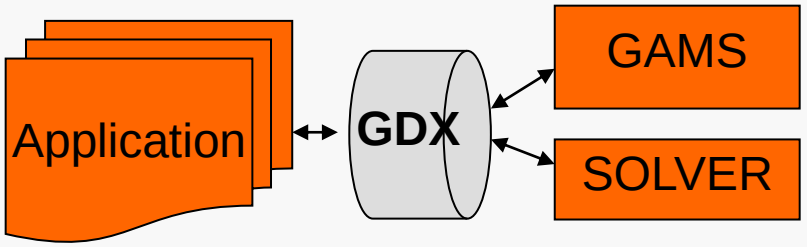
- Data transfer between different systems slow, error prone and bulky.
- Application (real time) require the capture of data instances that can be analyzed off-line in other environments.
- Management of name space mappings between different problems and their transformations into other data representations.
- Separate the model from its environment
- Search for a common low level high performance data container

GDX

Gams Data eXchange

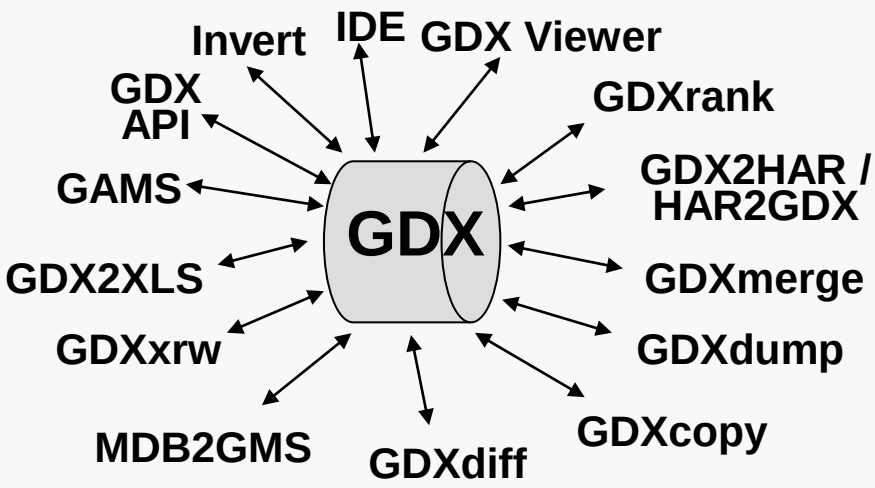
Gams Data eXchange

Binary Data Exchange

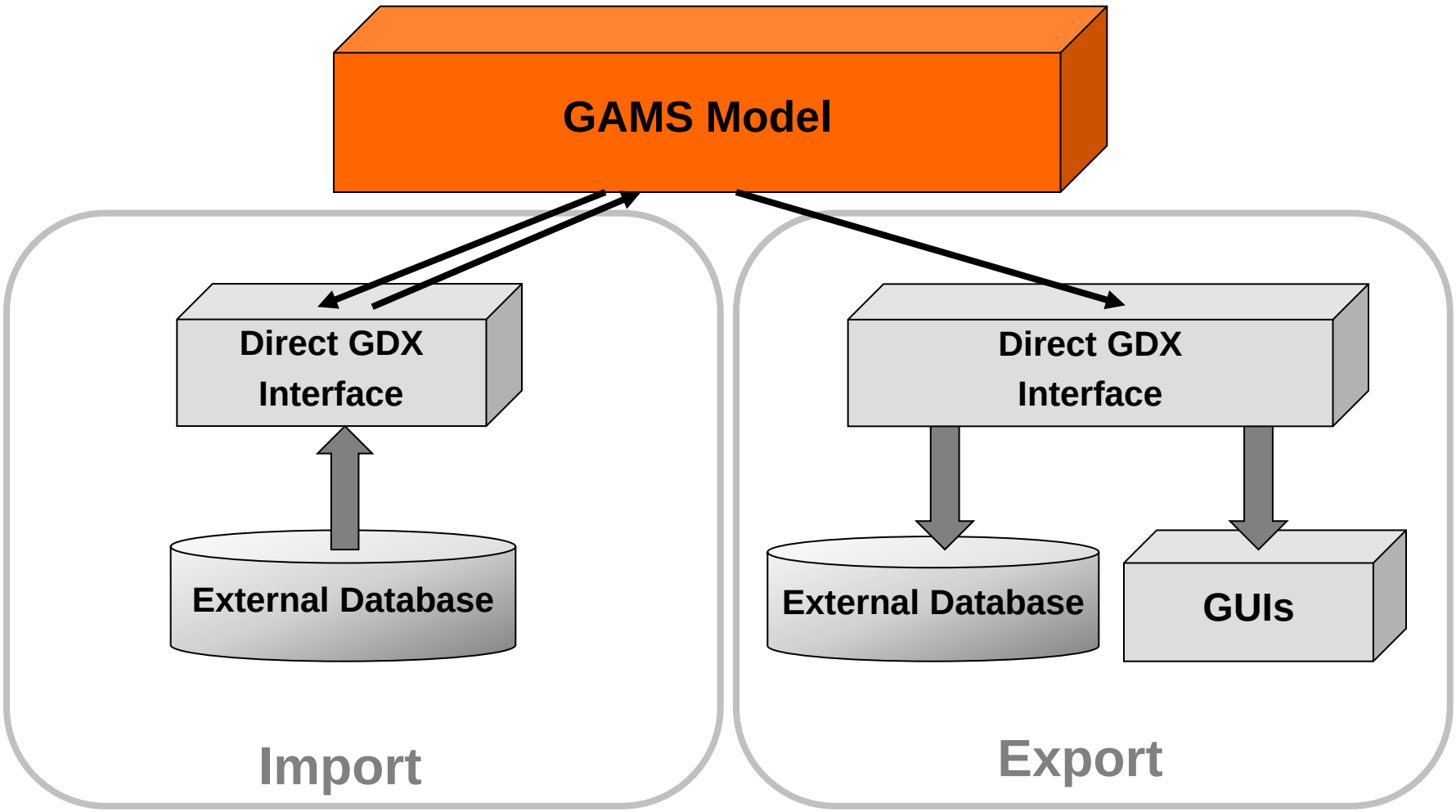


- Fast
- Safes disk space
- Tailored for large sparse matrices
- Platform independent
- Direct GDX interfaces
- API support for high-level programming languages
- Utilities

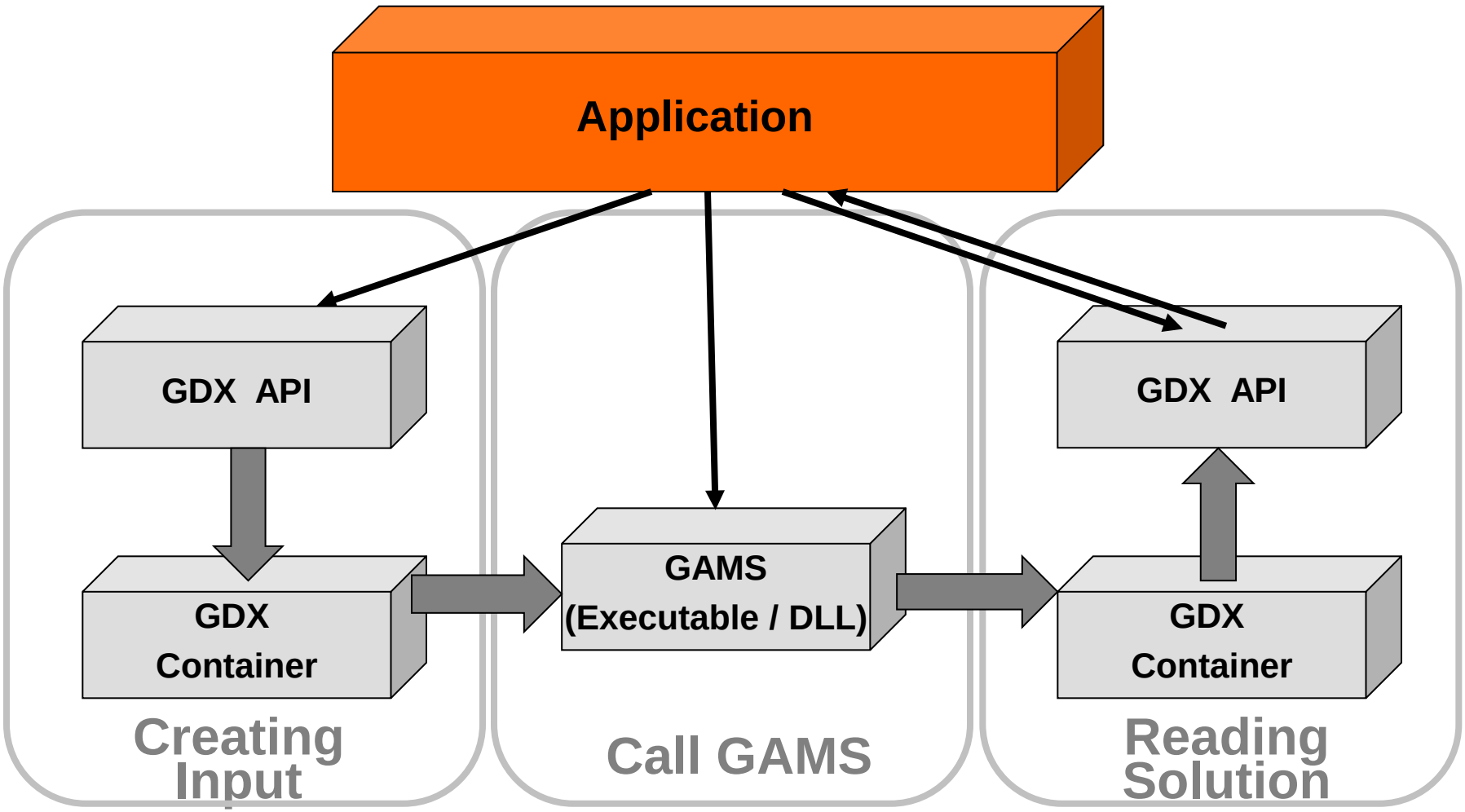
GDX Utilities



GAMS in Control

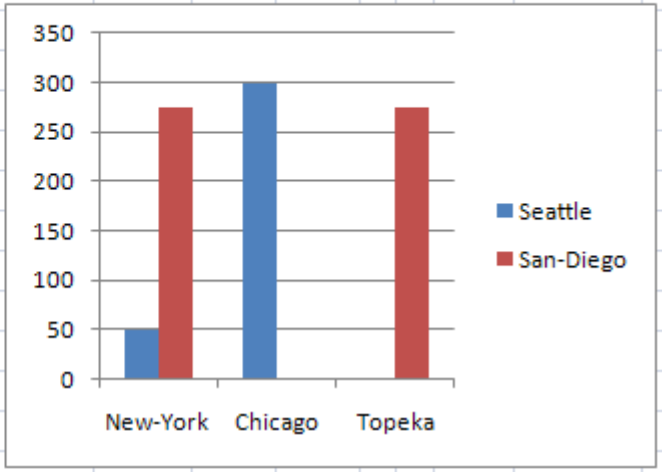


Application in Control



Incorporating a model in a spreadsheet

	A	B	C	D	E	F
1	**Distance**	New-York	Chicago	Topeka		Supply
2	Seattle	2.5	1.7	1.8		350
3	San-Diego	2.5	1.8	1.4		600
4						
5	Demand	325	300	275		
6						
7	Freight cost	90				
8						
9						
10	**SHIPMENT**	New-York	Chicago	Topeka		
11	Seattle	50	300	0		
12	San-Diego	275	0	275		



Clear Solution

GAMS Directory:	c:\program files\GAMS23.2\
Working Directory:	c:\tmp\
Solver:	CPLEX

SOLVE LP

SOLVE MIP

Solver: CPLEX
Equations: 6 Variables: 7
Model Status: 1 Optimal
Solver Status: 1 Normal Completion
Iterations: 4 Solve Time: 0.00

Objective Value: 153.675

Problem 2

- Building and maintaining solver specific links in different programming languages became a huge resource sink and made the introduction of new features difficult.

  → Simplify the building and maintaining of solver links
  → Manage multiple interacting models
  → Minimize the solver specific tailoring
  → Maintain one source only
  → Wrap automatically for different languages.

- Share libraries between the data management part of a modeling system and the solver. Example: function evaluations, first and second order derivatives, intervals,..

- Ease linking of experimental (meta-)solvers to GAMS

GMO

Gams Modeling Object

Gams Modeling Object (GMO)

GAMS' Next-Generation Model API

- Why a new model API?
- What do we need it to do?
- What does it look like? How is it put together?
- How did we do it?
- When are we going to be finished?

Solver Links – Different Perspectives

GAMS User

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

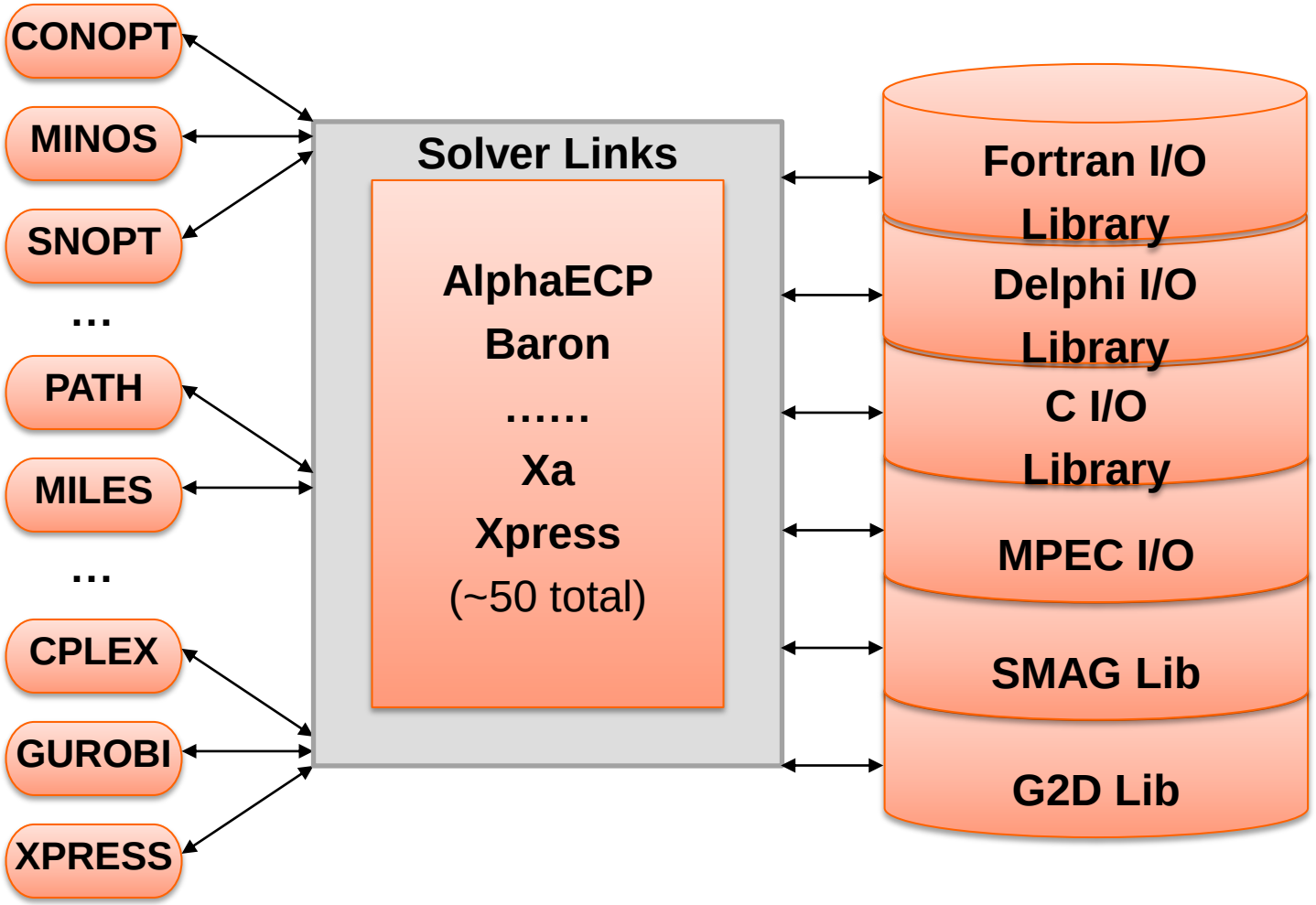
…nothing will change

Solver & Solver-link Developer

- IO Library provides access to
  - Matrix
  - Function/Gradient/Hessian evaluations
  - Solution file writer
  - Output handling
  - GAMS Options (e.g. resource limit)
  - Problem attributes (SOS, semicont, semiint, priorities, scales)
  - Utility routines
  - problem rewriting, matrix reordering

…our focus here

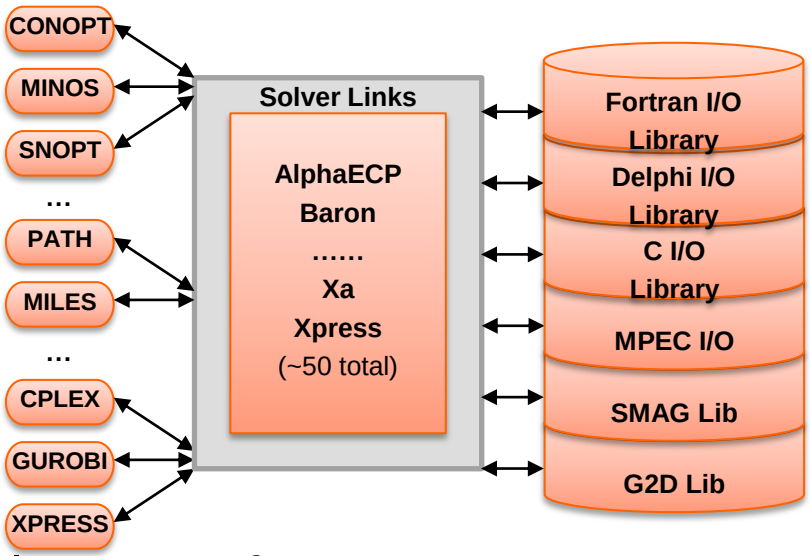
Reuse? What's that?!?



Multiple I/O Libraries - Advantages

Advantages

- proven over many years
- all platforms supported
- all GAMS-features available

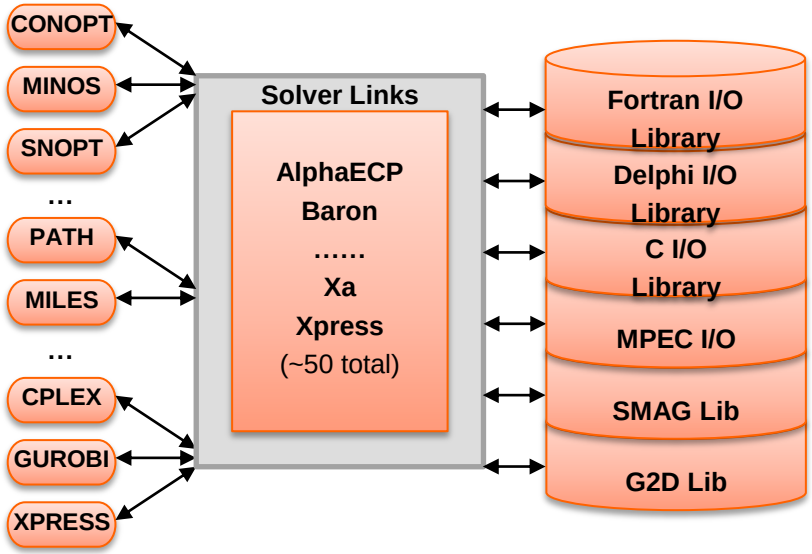


- written by language experts, use all language features
- resulted in high-quality links across solvers and platforms
  → has been one factor in our success

Multiple I/O Libraries - Disadvantages

Disadvantages

- Not always intuitive to use
  Linking your Solver to GAMS - THE COMPLETE NOTES (160 pages !!)
- Outdated design – I/O, STOP



- feature-poor (e.g. no automatic reformulation of objective func/var)
- inconvenient & expensive to maintain
- painful to move ‘inert mass’ forward
- linking your solver (without buddy at GAMS) is very difficult

Philosophy behind GMO

- **Then**
  - Computing environment: limited time and memory
  - Algorithm APIs not uniform or language-neutral
  - Expert users who understand optimization
  - *Don’t use unnecessary space or time*
  - *If the link gets in trouble, just abort, the user will fix things up and re-run.*
- **Now**
  - Most users won’t hit space/time limits
  - APIs look similar, are language-neutral
  - Users may be domain experts, not MP experts
  - *Use of additional space & time to give the GMO and GAMS user a better experience is justified.*

Checklist for GMO

- Powerful & convenient API – a few calls do the job
- In-core communication between GAMS and the solver, making potentially large model scratch files unnecessary
- Implement once, run everywhere (multiple platforms & multiple languages)
  - Platform-independent code, isolate the “dirty bits”.
  - API wrapper & multi-language interface
- Support meta-solvers (e.g. DICOPT, SBB, Examiner)
- Separate Model from Environment
- Comprehensive – one-stop shop for all linking needs
- Support shared-library implementation of solver links
- Support multiple models

Checklist for GMO

- Powerful & convenient API – a few calls do the job
- In-core communication between GAMS and the solver, making potentially large model scratch files unnecessary
- Implement once, run everywhere (multiple platforms & multiple languages)
  - Platform-independent code, isolate the “dirty bits”.
  - API wrapper & multi-language interface
- Support meta-solvers (e.g. DICOPT, SBB, Examiner)
- Separate Model from Environment
- Comprehensive – one-stop shop for all linking needs
- Support shared-library implementation of solver links
- Support multiple models

GMO: Powerful and Convenient API

- What's a powerful call?
  - Basic CS: information hiding, encapsulation, object model, abstraction
  - One call to do the job required, e.g. Hessian setup
  - No preconditions, magic calls, or nasty side effects

- Convenience - multiple routines and "flavors"
  - Jacobian row- vs. column-wise, tuples
  - Objective reformulation – function or variable
  - Free rows - yes or no
  - Column evals: dense or sparse, all or just NL
  - Common/typical tasks done in GMO, not the link

Checklist for GMO

- Powerful & convenient API – a few calls do the job
- In-core communication between GAMS and the solver, making potentially large model scratch files unnecessary
- Implement once, run everywhere (multiple platforms & multiple languages)
  - Platform-independent code, isolate the “dirty bits”.
  - API wrapper & multi-language interface
- Support meta-solvers (e.g. DICOPT, SBB, Examiner)
- Separate Model from Environment
- Comprehensive – one-stop shop for all linking needs
- Support shared-library implementation of solver links
- Support multiple models

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
  - 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)

***** without Virus Scanner

Checklist for GMO

- Powerful & convenient API – a few calls do the job
- In-core communication between GAMS and the solver, making potentially large model scratch files unnecessary
- Implement once, run everywhere (multiple platforms & multiple languages)
  - Platform-independent code, isolate the “dirty bits”.
  - API wrapper & multi-language interface
- Support meta-solvers (e.g. DICOPT, SBB, Examiner)
- Separate Model from Environment
- Comprehensive – one-stop shop for all linking needs
- Support shared-library implementation of solver links
- Support multiple models

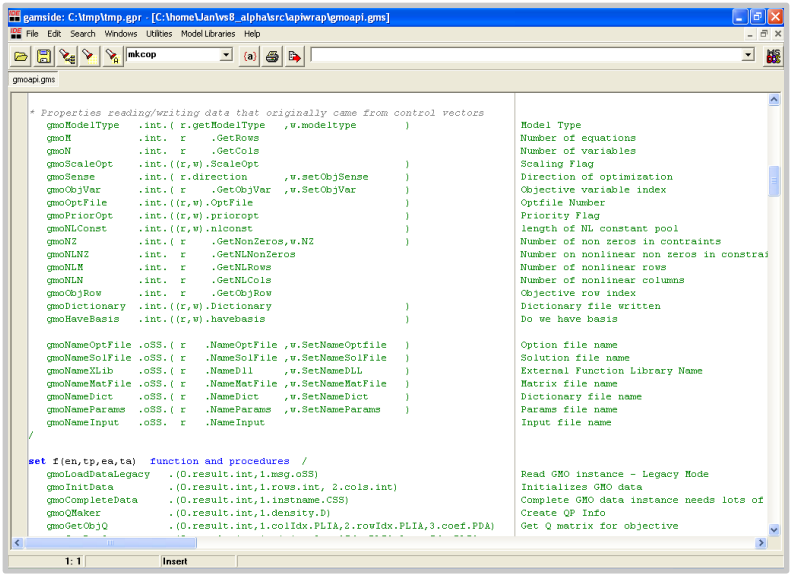
Implement Once, Run Everywhere

- All GMO coding done in a single language and style
  - Allows code sharing with other components
  - Allows for shared development (GMO is a team effort)
- All GMO coding is platform-independent
  - Makes writing code faster, more reliable
  - Maintenance is simplified
- Platform-dependent code isolated in utility libraries
  - Makes adding a new platform easier
  - Maintenance is simplified
  - Unit testing is easy and effective

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



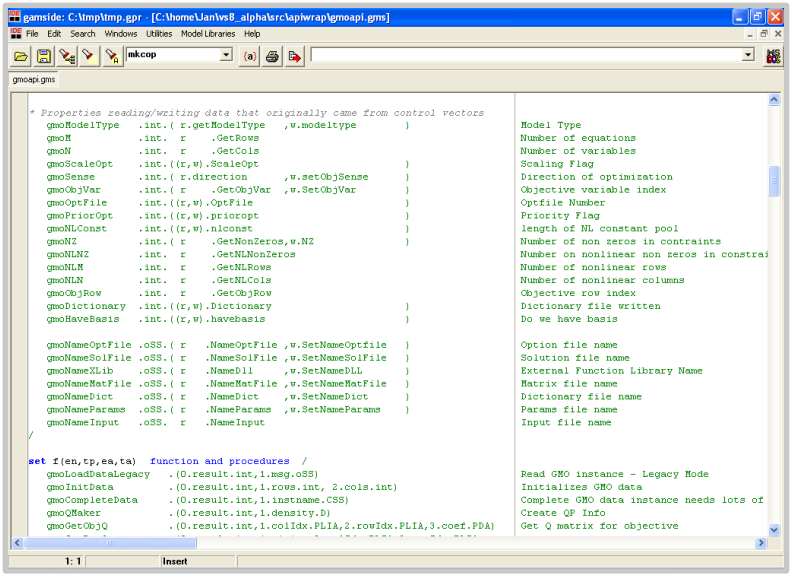
→ A change in the definition of the API immediately makes it into all language interfaces

→ No manual and therefore error-prone efforts required

Automated Generation of APIs

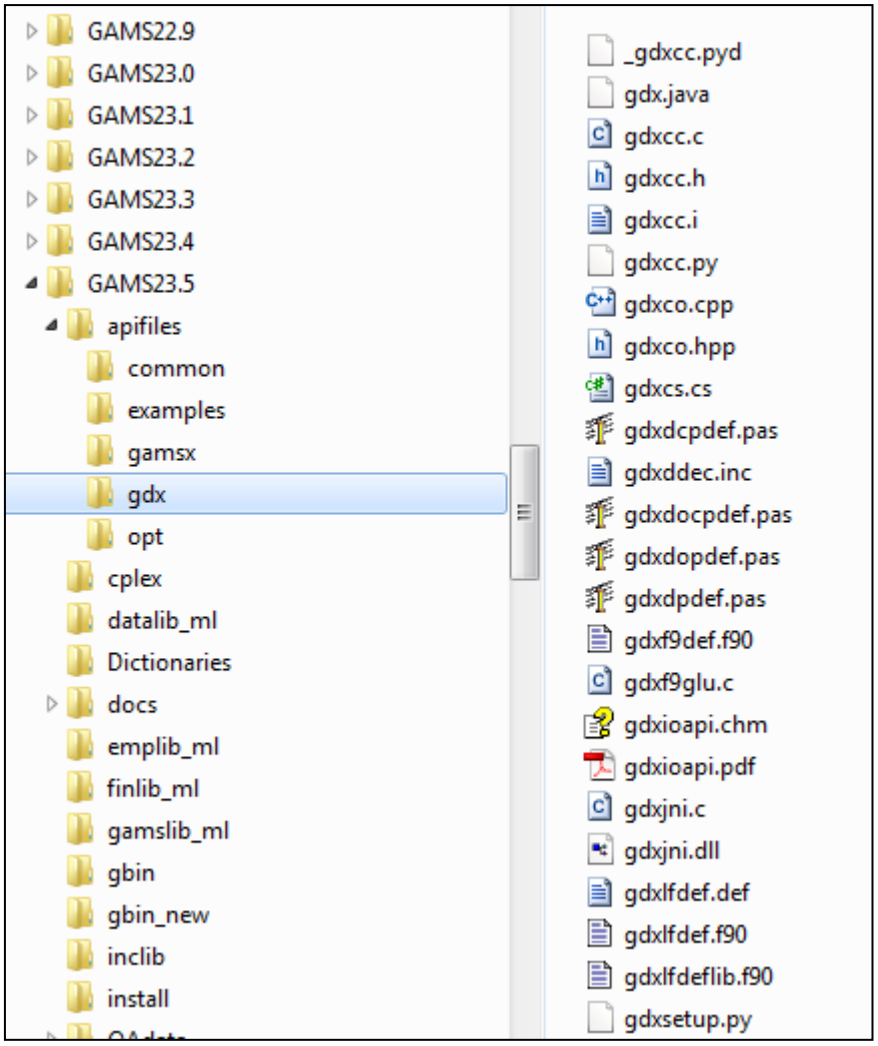
‘The GAMS Wrapper’

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



Distributed GAMS APIs

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

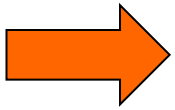


Checklist for GMO

- Powerful & convenient API – a few calls do the job
- In-core communication between GAMS and the solver, making potentially large model scratch files unnecessary
- Implement once, run everywhere (multiple platforms & multiple languages)
  - Platform-independent code, isolate the “dirty bits”.
  - API wrapper & multi-language interface
- Support meta-solvers (e.g. DICOPT, SBB, Examiner)
- Separate Model from Environment
- Comprehensive – one-stop shop for all linking needs
- Support shared-library implementation of solver links
- Support multiple models

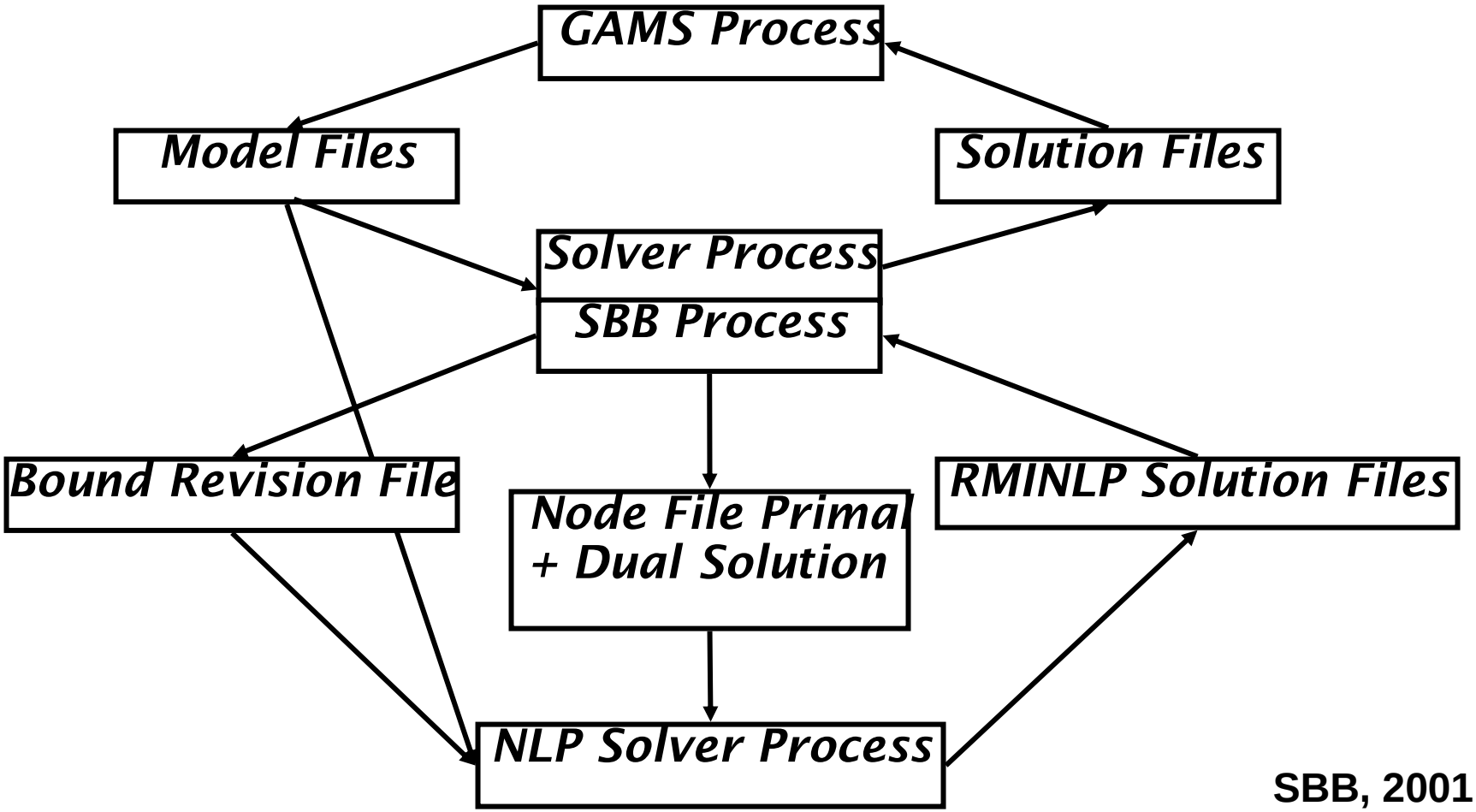
Meta-Solvers with GMO

- Populated GMO object (e.g. by GAMS)
- GMO API to allow modification and alteration of bounds, rhs, “modifiable” parameters (NL expression evaluation)
- GMO/GEV (GAMS Environment Object) based solver links
- Runtime system (C, Python, Java, …)



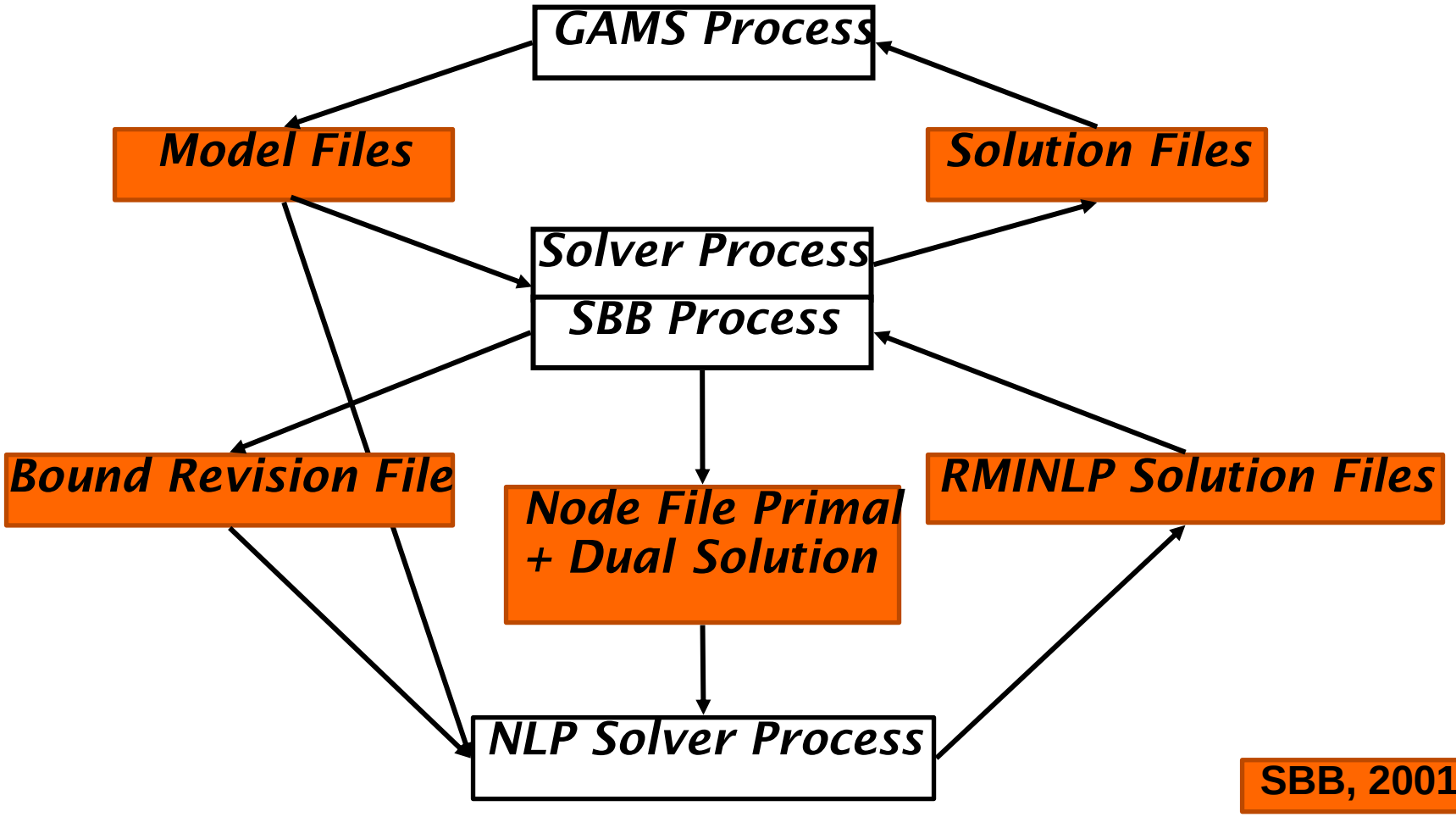
- Alternative way to implement decomposition, and other algorithmic ideas based on MP models

‘Efficient’ Implementation of B&B

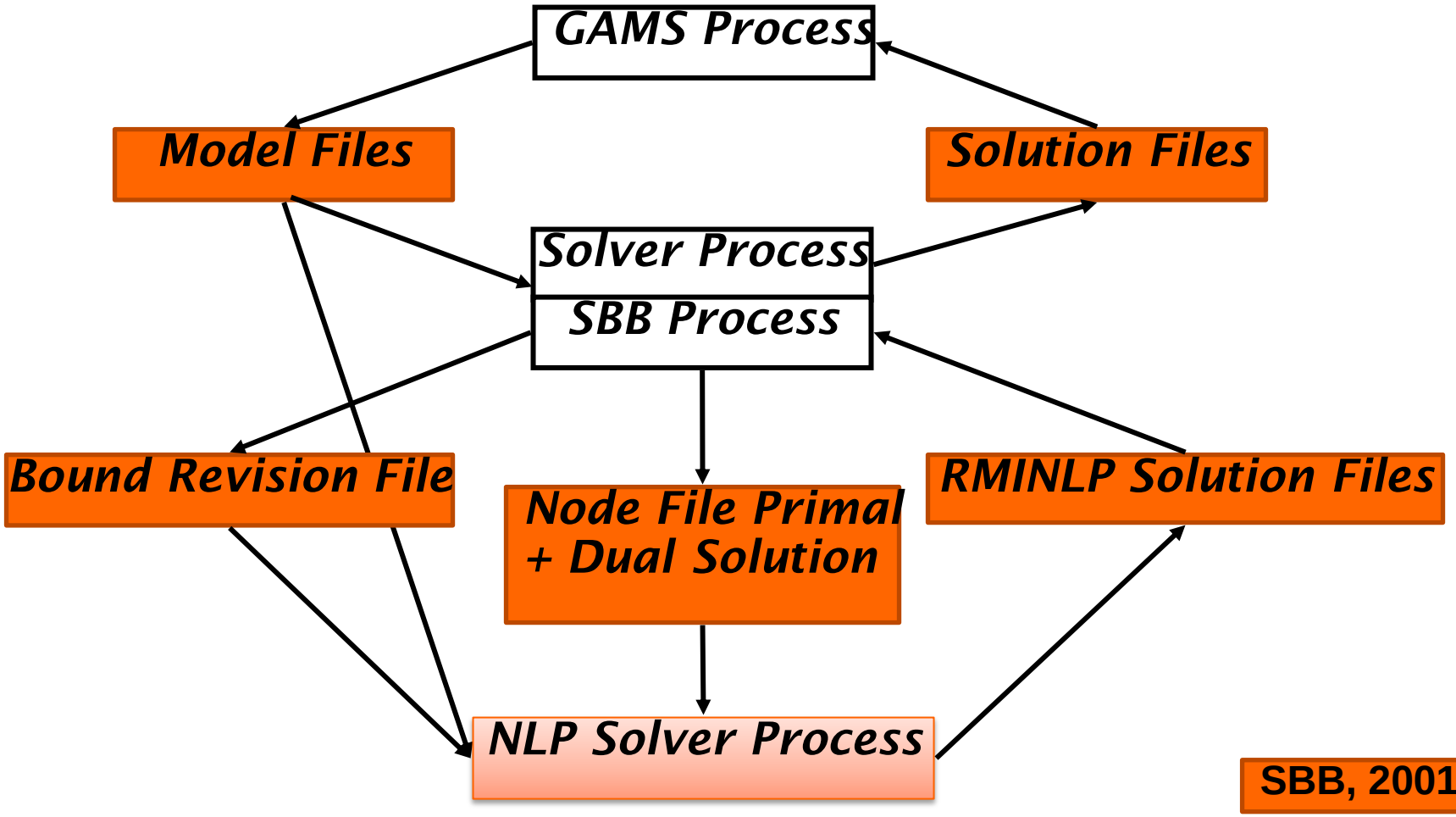


SBB, 2001

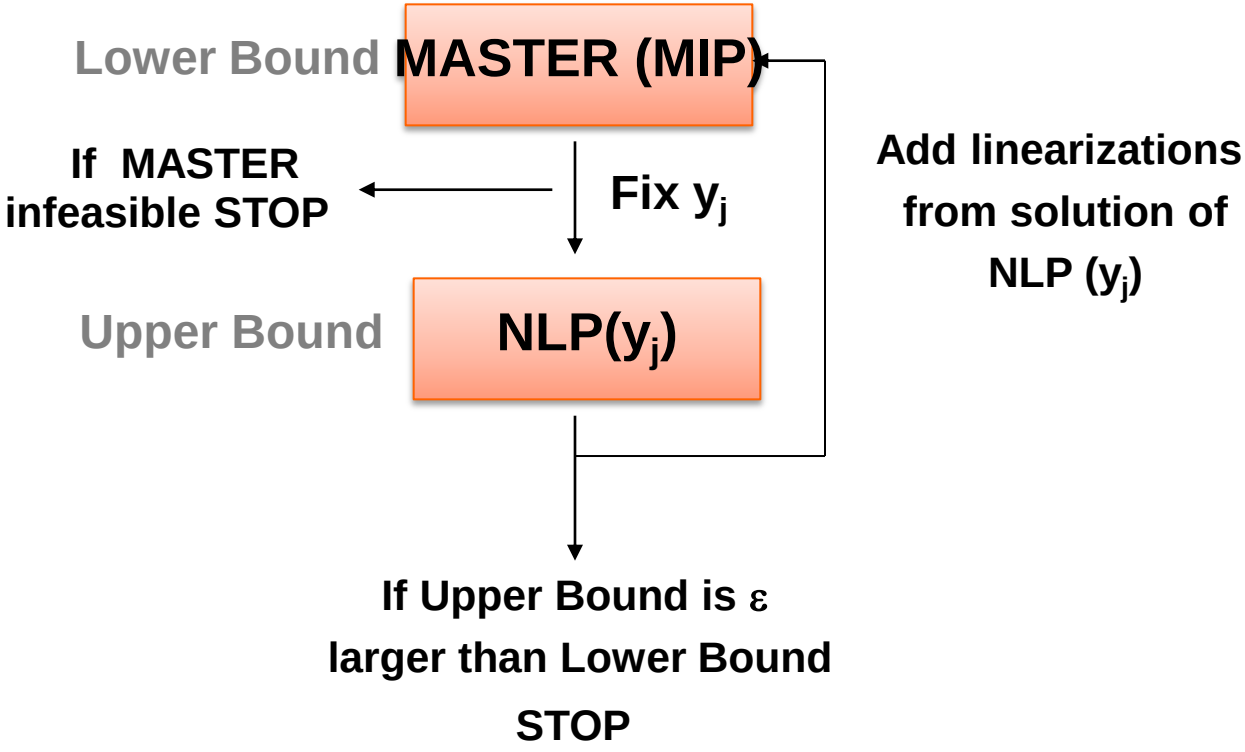
‘Efficient’ Implementation of B&B



‘Efficient’ Implementation of B&B



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

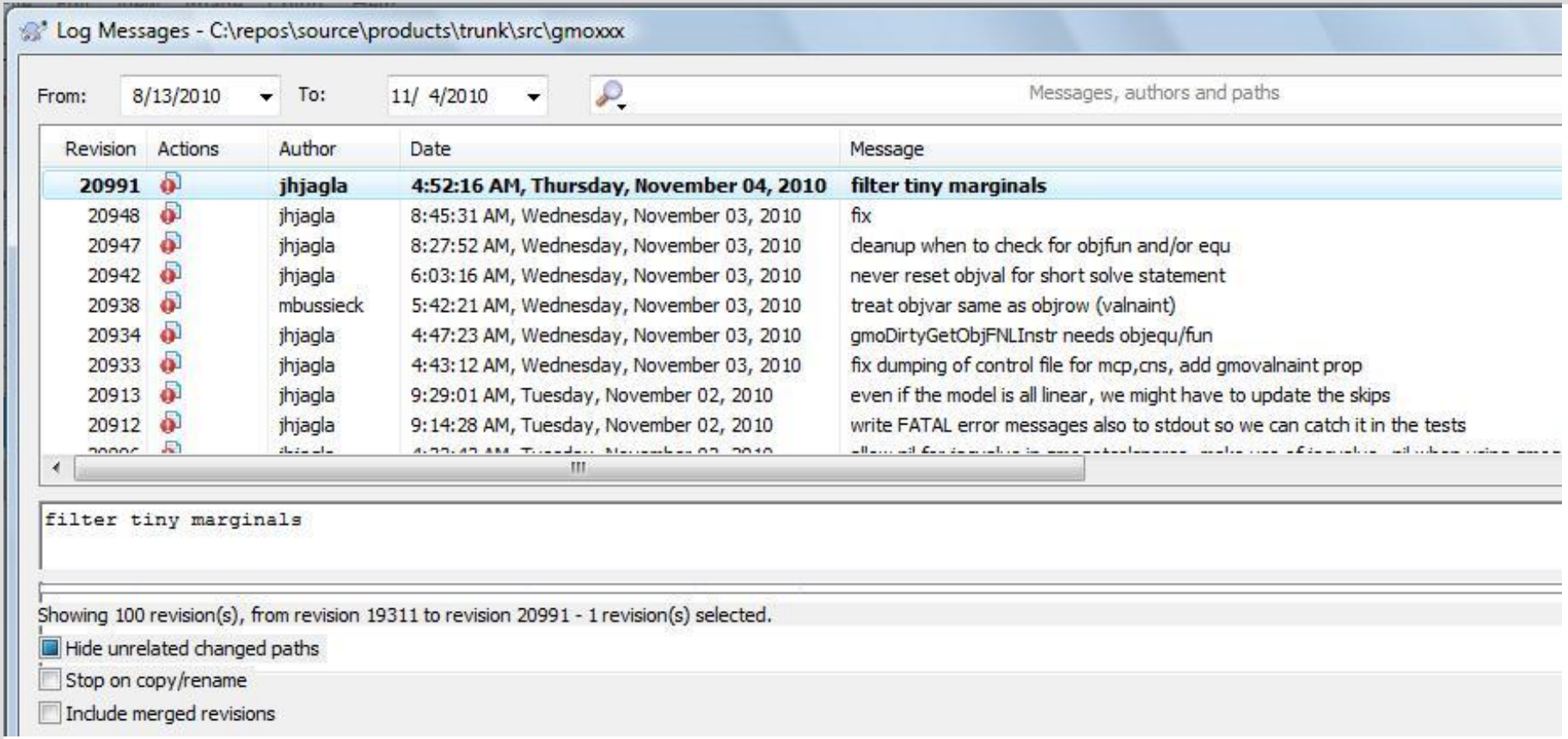
- Basically start a whole new process over and over

New Dicopt Implementation

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

- Object Oriented
- Use C++ Interface to GMO
- Two models: NLP and MIP
- Use standardized solver interface to call NLP/MIP solver in-core (pass GMO ‘handle’ on to solver)
- Algorithmic improvements

How We Did It: SVN



- Everybody agrees: Version control is a game-changer

How We Did It: Automation & Testing

GAMS

[Home | Support | Sales | Solvers | Documentation | Model Library | Search | Con

Latest GAMS System Builds and Test Results

Tuesday 09Nov10 15:

[Latest Builds | Alpha Builds | Beta Builds | Nightly Builds | System Codes | History]

nightly α	System	Libraries	Build	Rev	Status and Time (UTC)		Initial Tests		Full Tests
Monday	lnx	Download	23.7.0	21065	Test done	09Nov2010 06:52:24	712 runs 0 failures (q=0,s=0)	Report	8902 runs 2 failures (q=1,s=1)
Monday	lx3	Download	23.7.0	21065	Test done	09Nov2010 09:11:56	732 runs 0 failures (q=0,s=0)	Report	9385 runs 2 failures (q=1,s=1)
Tuesday	vs8	Download	23.7.0	21075	Test done	09Nov2010 14:32:06	734 runs 0 failures (q=0,s=0)	Report	9397 runs 2 failures (q=1,s=1)
Monday	wei	Download	23.7.0	21072	Test done	09Nov2010 07:05:24	682 runs 1 failures (q=1,s=0)	Report	8364 runs 5 failures (q=2,s=3)

nightly β	System	Libraries	Build	Rev	Status and Time (UTC)		Initial Tests		Full Tests
Monday	lnx	Download	23.6.0	21070	Test done	09Nov2010 07:22:45	712 runs 0 failures (q=0,s=0)	Report	8901 runs 2 failures (q=1,s=1)

How We Did It: Automation & Testing

- SVN and other tools automate builds on all platforms

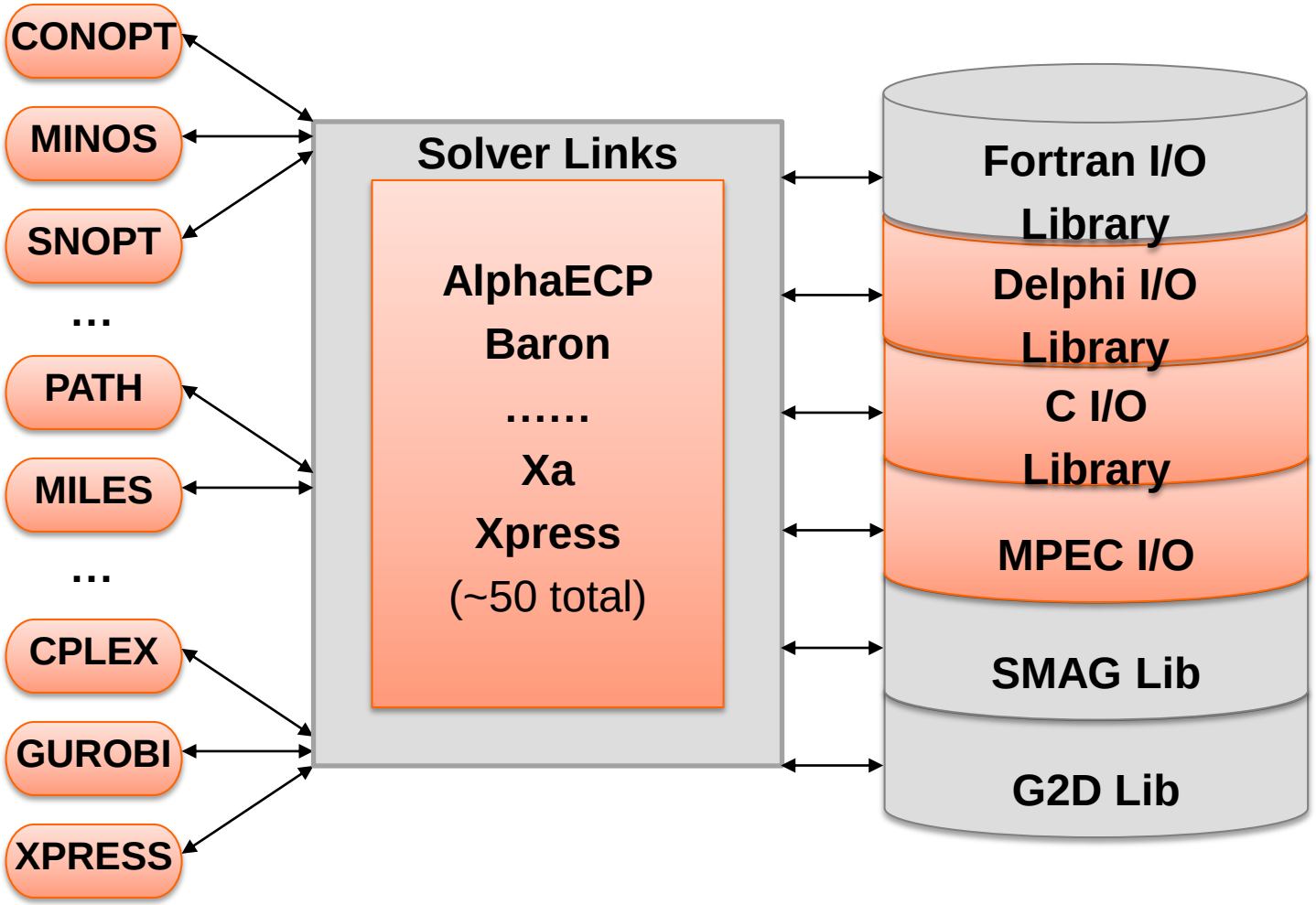
- Extensive, automated tests
  - Test library (503 models), other libraries (hundreds)
  - Runs over all solvers, some NLP/MIP combinations
  - Recent beta: 17 test machines, each ~ 3K – 10K
  - Collecting, archiving, sharing of test results

- PAVER used to compare to previous versions
  - Helps find outliers (bugs), problem cases
  - http://www.gamsworld.org/performance/paver/

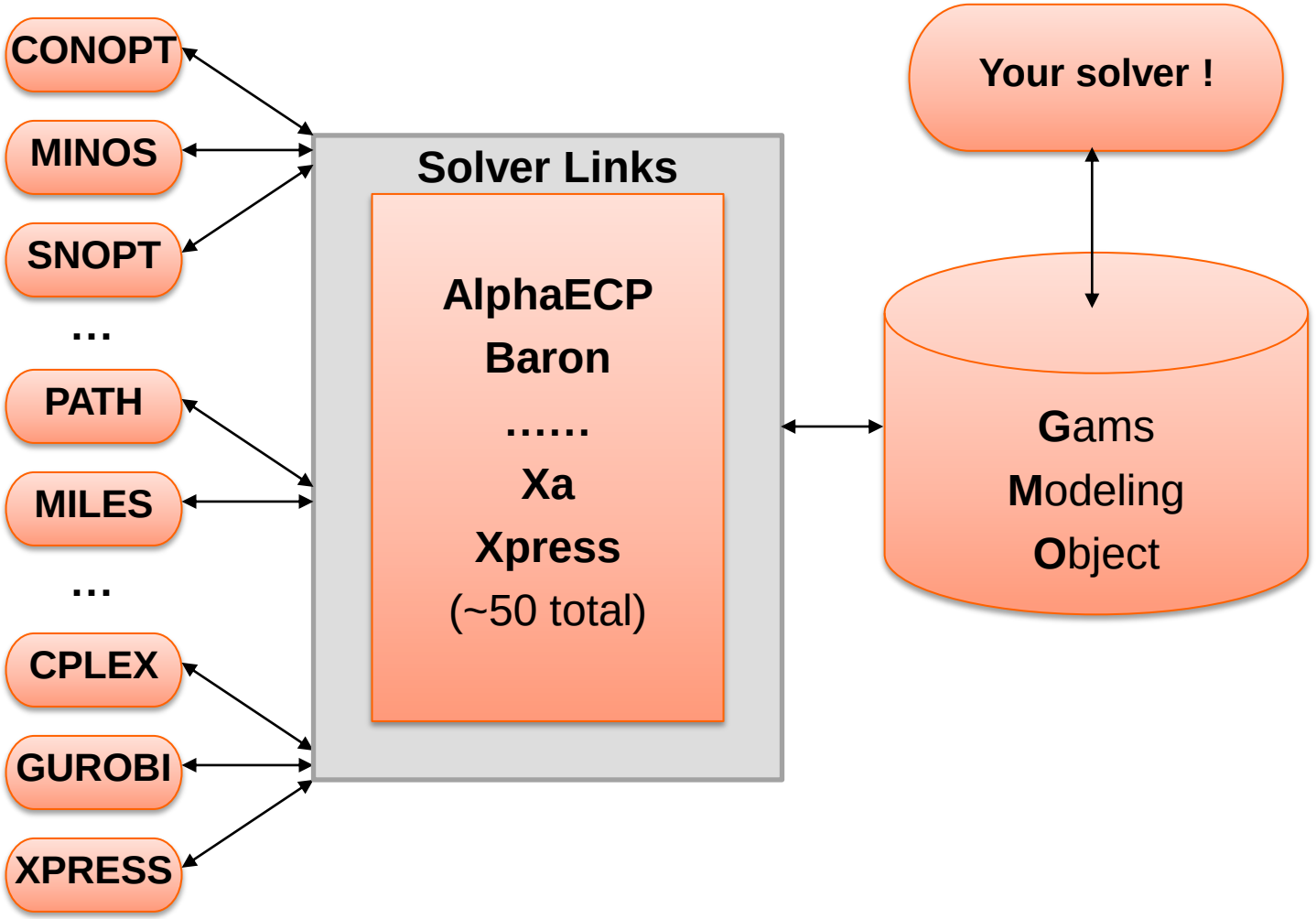
When Will We Be Finished?

- **GAMS 23.5.2 (current distribution)**
  - Coin-Solvers (Bonmin, Cbc, Couenne, Ipopt, OS)
  - Gurobi
  - Lindoglobal
  - OSI-based links to Cplex, Gurobi, Glpk, Mosek, Xpress
  - Scip
  - …
- **GAMS 23.6 (currently in beta)**
  - All previous Fortran links (e.g. Conopt, Minos, Snopt)
  - All links using 2nd-order information (e.g. Knitro, Pathnlp, Mosek)
  - Xpress
- **GAMS 23.7 ??**

Summary


CONOPT
MINOS
SNOPT
…
PATH
MILES
…
CPLEX
GUROBI
XPRESS
Solver Links
AlphaECP
Baron
……
Xa
Xpress
(~50 total)
Fortran I/O Library
Delphi I/O Library
C I/O Library
MPEC I/O
SMAG Lib
G2D Lib

Summary


CONOPT
MINOS
SNOPT
…
PATH
MILES
…
CPLEX
GUROBI
XPRESS
Solver Links
AlphaECP
Baron
……
Xa
Xpress
(~50 total)
Your solver !
Gams
Modeling
Object

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, more agile
- GMO opens up new possibilities for moving GAMS forward
- GMO interfaces are not yet public but alpha version can be made available on request

Scenario Solver

Use GDX and GMO

GAMS Scenario Solver

```
Loop(s,
     d(i,j) = dd(s,i,j);
     f = ff(s);
     solve mymodel min z using lp;
     rep(s) = mymodel.objval;
);
```

Setting	Solve time (secs)
Solvelink=0 *(default)*	40.297
Solvelink=%Solvelink.LoadLibrary%	03.625

GAMS Scenario Solver

```
cost.. z=e=sum((i,j),f*d(i,j)/1000*x(i,j));
set dict / s.scenario.''
           d.param   .dd
           f.param   .ff
           x.level   .xx /
solve mymodel min z using lp scenario dict;
```

- Update model data instead of matrix coefficients/rhs
- Hot start (keep the model hot inside the solver and use solver's best update mechanism)
- Save model generation and solver setup time
- Model rim unchanged from scenario to scenario
- Apriori knowledge of all scenario data

Problem 3

- Problems may contain
  - Complementarity
  - Hierarchy
  - Interacting agents
  - Risk measures
  - Logic relationships
- Cannot be expressed with current modeling languages and have no direct solution method.
- Example: General equilibrium models are a transformation from multi agent optimization/variational problems into a single mixed complementarity model.
- How to automate the transformations by annotations of existing optimization models that convey model structure to the solver.

EMP

Extended Math. Programming

Current state: Model-Side

- Traditional problem format

$$\min_{x} c(x) \quad s.t.\ A_1(x) \le b_1,\ A_2(x) = b_2$$

- Support for complementarity constraints
- Interactions between models possible
  - Series of models
  - Scenario analyses / parallelized model runs
  - Iterative sequential feedback
  - Decomposition

New solution concepts

- Embedded Complementarity Systems
- Disjunctive Programs
- Bilevel Programs
- Extended Nonlinear Programs
- Variational Inequalities
- …

  - Breakouts of traditional MP classes
  - No conventional syntax
  - Limited support with common model representation
  - Incomplete/experimental solution approaches
  - Lack of reliable/any software

What now?

Do not:

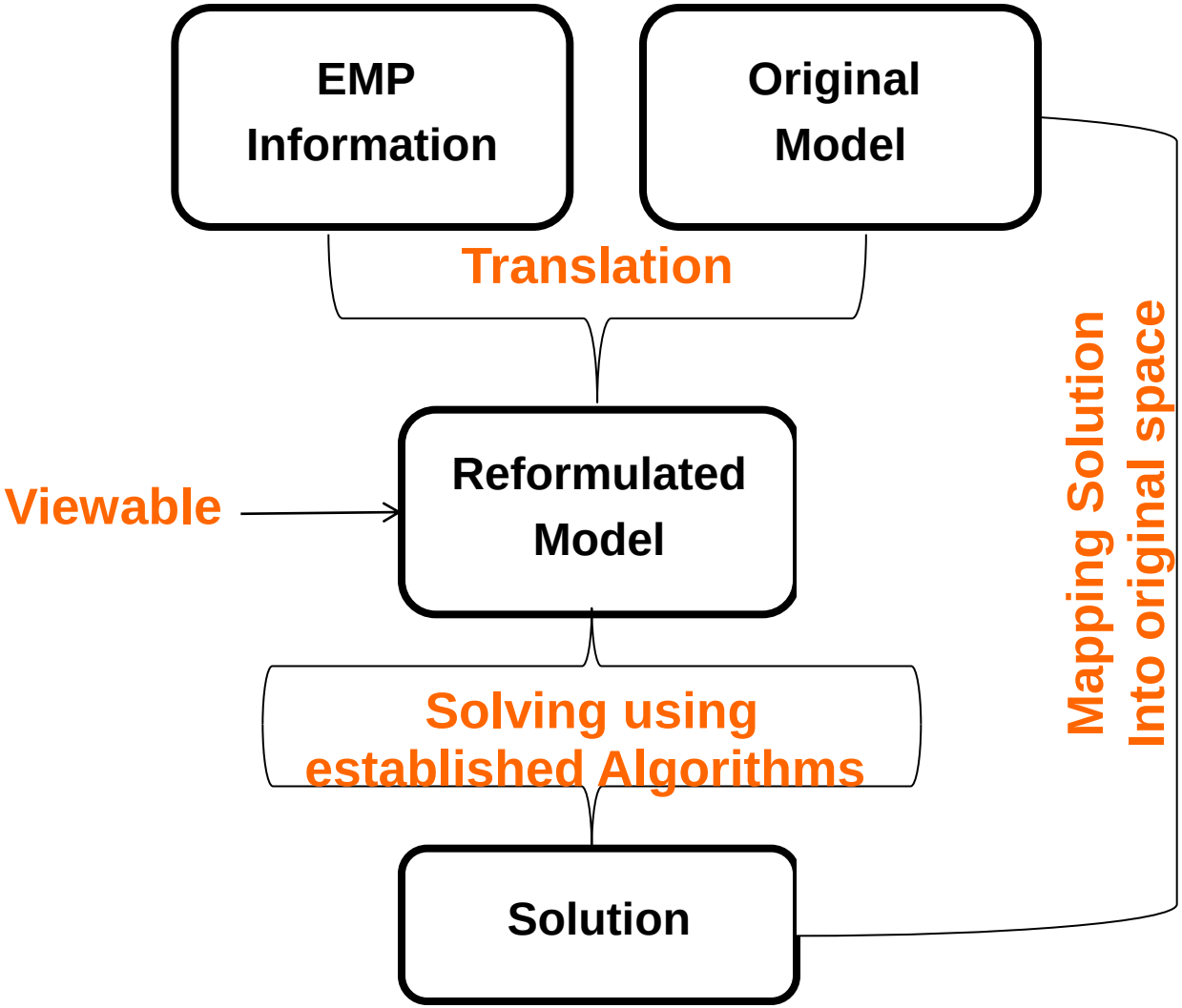
- overload existing GAMS notation right away !
- attempt to build new solvers right away !

But:

- Use existing language features to specify additional model features
- Distribute information as part of the production system
- Express extended model in symbolic form and apply existing matured solution technology

→ **Extended Mathematical Programming (EMP)**

GAMS "Solver" EMP



Bilevel Programming

$$\begin{aligned} \min_{x,y} \quad & f(x,y) \\ \text{s.t.} \quad & g(x,y) \leq 0, \\ & y \text{ solves } \min_{s} v(x,s) \text{ s.t. } h(x,s) \leq 0 \end{aligned}$$

- Additional Information:

```
$onecho  > %emp.info%
Bilevel x min v h
$offecho
```

- EMP Tool automatically creates an MPEC by expressing the lower level optimization problem through its optimality conditions

Bilevel Model

Conejo A J, Castillo E, Minguez R, and Garcia-Bertrand R; Decomposition Techniques in Mathematical Programming, Springer,Berlin, 2006.

```
defobj.. z =e= sqr(x1+x2-2) + sqr(x3+x4-2);
a0.. h1 =e= h1_0;
a1.. h2 =e= h2_0;
a2.. x1-x2 =e= 3;
```

Outer Problem

```
defh1.. h1 =e= sqr(u1-x1) + sqr(u2-x2) + sqr(u3-x3) + sqr(u4-x4);
e1.. 3*u1 + u2 + 2*u3 + u4 =e= 6;
```

Inner Problem 1

```
defh2.. h2 =e= sqr(v1-x1) + sqr(v2-x2) + sqr(v3-x3) + sqr(v4-x4);
e2.. v1 + v2 + v3 + 2*v4 =e= 7;
```

Inner Problem 2

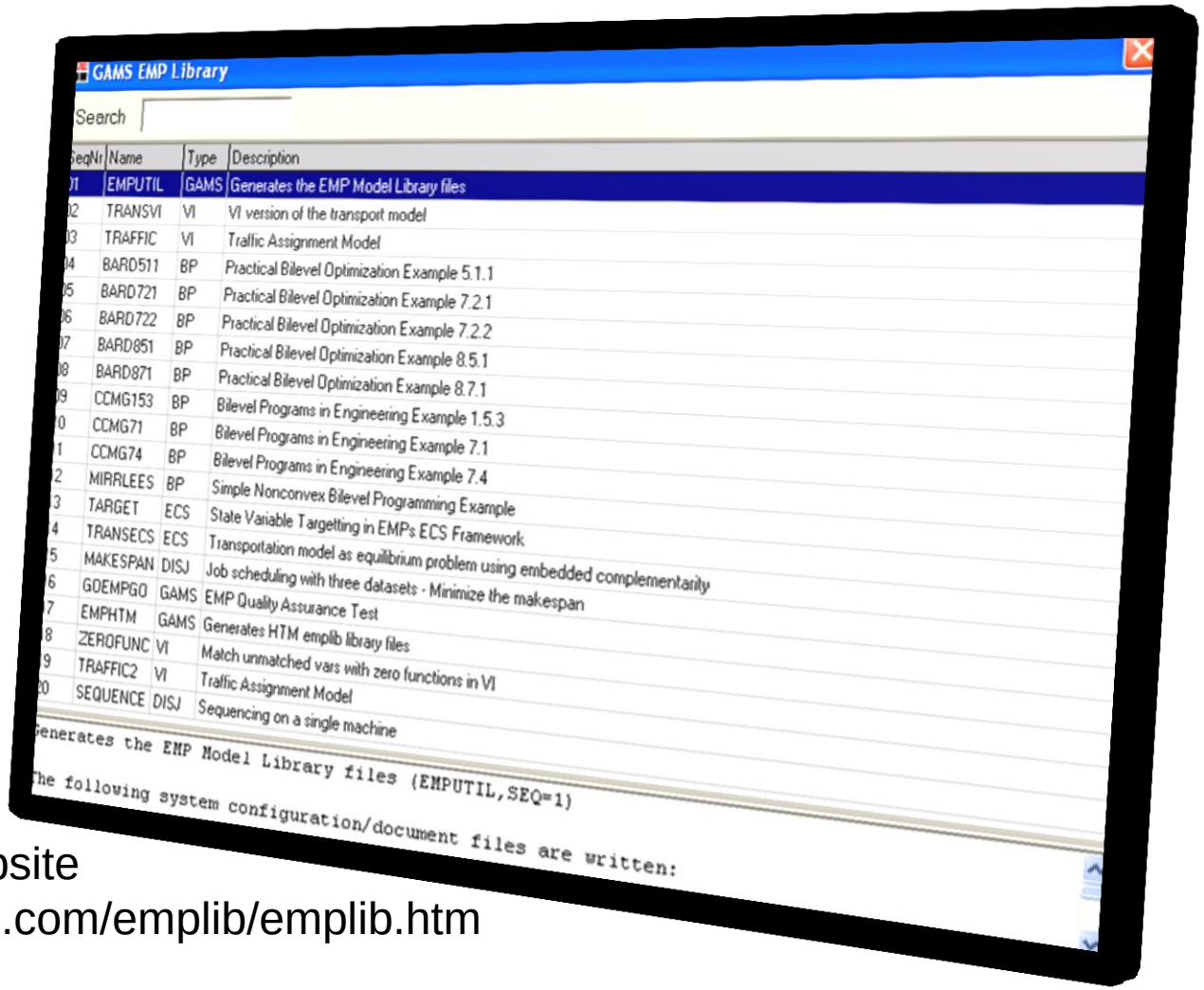
EMP Information File + EMP Summary Log

```
model emp / master, submodel1, submodel2 /;

$onecho > "%emp.info%"
bilevel x1 x2 x3 x4
min h1 * defh1 e1
min h2 * defh2 e2
$offecho

solve emp us emp min z;
```

```
JAMS - Solver for Extended Mathematical Programs (EMP)
------------------------------------------------------
--- EMP Summary  (errors=0)
    Adjusted Constraint = 0
    Flipped Constraints = 0
    Dual Variable Maps  = 0
    Dual Equation Maps  = 0
    VI Functions        = 0
    Equilibrium Agent   = 0
    Bilevel Followers   = 2
    Disjunctions        = 0
    The model C:\tmp\22fe\emp.dat will be solved by GAMS
```

EMP Library

- Distributed with GAMS
- Available on website http://www.gams.com/emplib/emplib.htm



Summary

EMP

- automates symbolic reformulations to avoid error-prone and time-consuming manual algebra (re)writing
- offers solutions where solutions couldn't be offered before
  - Embedded Complementarity Systems
  - Disjunctive Programs
  - Bilevel Programs
  - Extended Nonlinear Programs
  - Variational Inequalities
- facilitates to compare concurrent strategies
- free

- But: non-exhaustive, yet!

BETA 23.6

GAMS 23.6 Beta

Released November, 6th

www.gams.com/beta

- New Solver Libraries
  - COIN-OR (BONMIN 1.4, CSDP 6.1.1, …)
  - CPLEX 12.2.0.1
  - GUROBI 4.0
  - KNITRO 7.0
  - MOSEK 6.0.91
  - SCIP 2.0
  - XPRESS 21.01

- More solvers support in-core communication
  - Conopt, Knitro, Lgo, Mosek, Xpress

GAMS 23.6 Beta cont'd

Released November 6th

www.gams.com/beta

- Chk4Upd
- Python APIs to component libraries
- GAMSIDE updates
- XLSDump
- New library models (datalib, emplib, modlib, testlib)
- Internal Reorganization (non-linear instructions, hessian evaluators)

Contacting GAMS

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

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
support@gams-software.com