GAMS

Stefan Vigerske
stefan@gams.com

October 2nd, 2015, CO@Work, Berlin

Prologue

Material to this lecture: `http://co-at-work.zib.de/files/gams/`

- **CO@Work virtual machines**: GAMS is installed (run `gams`)
- **Download** GAMS system: `http://www.gams.com/download`
- **Evaluation license** (valid until 7.1.2016):
  `http://co-at-work.zib.de/files/gams/gamslice.txt`
  (already installed on 32bit VM)

Prologue

Material to this lecture: http://co-at-work.zib.de/files/gams/

- **CO@Work virtual machines**: GAMS is installed (run gams)
- **Download** GAMS system: http://www.gams.com/download
- **Evaluation license** (valid until 7.1.2016):
  http://co-at-work.zib.de/files/gams/gamslice.txt
  (already installed on 32bit VM)

Next 85 minutes:

- A short overview on GAMS.
- ~~GAMS syntax and programming.~~ (gamsbasics.pdf)
- Equitable coach problem in GAMS.
- Exercise: Run and modify equitable coach problem.
- "Hiding GAMS": Calling GAMS from Python.
- Exercise: Cutting Stock by Column Generation in Python.

Company

- **Roots**: World Bank, 1976
- Went **commercial** in 1987
- **Locations**
  - GAMS Development Corporation (Washington, D.C.)
  - GAMS Software GmbH (Germany)

Washington

Braunschweig

Frechen (Cologne)

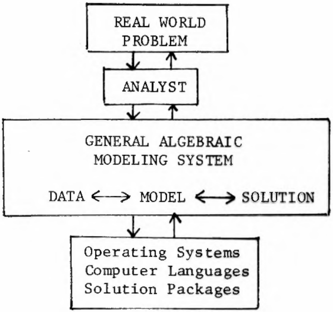
ZIB, Berlin

- $\approx$ **16 employees** (7 development, 5 sales, ...)
- **Product**: The **G**eneral **A**lgebraic **M**odeling **S**ystem

The Vision: World Bank Slide, 1976

GAMS

IDEAL TECHNOLOGY



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

Timeline: 1976 . . . now

1976	GAMS idea is presented at ISMP, Budapest
1978	Phase I: GAMS supports **linear programming**. Supports **Mainframes** and **Unix** Workstations
1979	Phase II: GAMS supports **nonlinear programming**
1987	GAMS becomes a commercial product
1988	First **PC System** (16 bit)
1988	Alex Meeraus, the initiator of GAMS and founder of GAMS Development Corporation, is awarded INFORMS Computing Society Prize
1990	**32 bit Dos Extender**
1990	GAMS moves to Georgetown, Washington, D.C.
1991	**Mixed Integer Non-Linear** Programs capability (DICOPT)
1994	GAMS supports **mixed complementarity** problems
1995	MPSGE language is added for CGE modeling
1996	European branch opens in Germany
1998	**32 bit native Windows**
1998	**Stochastic programming** capability (OSL/SE, DECIS)
1999	Introduction of the GAMS **Integrated development environment** (IDE)
2000	GAMS World initiative started
2001	**GAMS Data Exchange** (GDX) is introduced
2002	GAMS is listed in OR/MS 50th Anniversary list of milestones
2003	**Conic programming** is added
2003	**Global optimization** in GAMS
2004	Support for Quadratic Constrained programs
2005	Support for **64 bit PC** Operating systems
2006	GAMS supports parallel **grid computing**
2007	GAMS supports open-source solvers from COIN-OR
2008	Support for 32 and 64 bit **Mac OS X**
2009	GAMS supports extended mathematical programs (EMP)
2010	GAMS is awarded the company award of the German Society of Operations Research (GOR)
2012	The Winners of the 2012 INFORMS Impact Prize included Alexander Meeraus. The prize was awarded to the originators of the five most important algebraic modeling languages.

User Community

GAMS

- more than **11500 licenses**
- **50% academic** users, **50% commercial**

Application Areas:

- Agricultural Economics
- Chemical Engineering
- Econometrics
- Environmental Economics
- Finance
- International Trade
- Macro Economics
- Management Science/OR
- ...
- Applied General Equilibrium
- Economic Development
- Energy
- Engineering
- Forestry
- Logistics
- Military
- Mathematics

Language

Declarative Language:

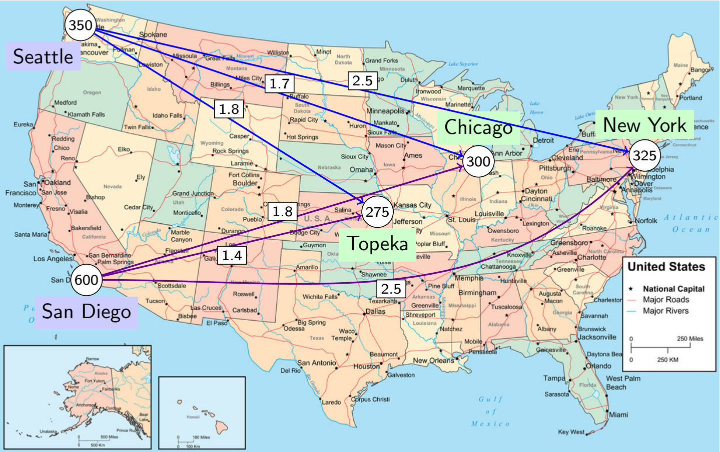
- Similar to **mathematical notation**
- Few **basic language elements**: sets, parameters, variables, equations, models
- Model is executable (algebraic) description of the problem

Imperative Elements:

- **Control flow** statements: loops, for, if, ...
- build algorithms within GAMS
- **exchange data** with other systems

Transport Example (GAMS Model Library: TRNSPORT)

Canning Plants (supply) —shipments (number of cases)→ **Markets** (demand)



Minimize	Transportation cost
subject to	Demand satisfaction at markets
	Supply constraints

Transport Example (GAMS Model Library: TRNSPORT)

GAMS

i	canning plants
j	markets
a_i	capacity of plants
b_j	demand at markets
$c_{i,j}$	transportation cost per case
$x_{i,j}$	cases to ship from i to j

$$
\begin{aligned}
\min \quad & \sum_{i,j} c_{i,j} x_{i,j} & \\
\text{s.t.} \quad & \sum_{j} x_{i,j} \le a_i & \forall i \\
& \sum_{i} x_{i,j} \ge b_j & \forall j \\
& x_{i,j} \ge 0 & \forall i,j
\end{aligned}
$$

```
Sets
  i  "canning plants",
  j  "markets";

Parameters
  a(i)   "capacity of plant i in cases",
  b(j)   "demand at market j in cases",
  c(i,j) "transport cost in 1000$ per case";

Variables
  x(i,j) "shipment quantities in cases",
  z      "total transportation costs in 1000$"

Positive Variable x;

Equations
  cost      "define objective function",
  supply(i) "observe supply limit at plant i",
  demand(j) "satisfy demand at market j";

cost ..       z =e= sum((i,j), c(i,j)*x(i,j));
supply(i) .. sum(j, x(i,j)) =l= a(i);
demand(j) .. sum(i, x(i,j)) =g= b(j);

Model transport / all / ;
```

GAMS is not a Solver!

GAMS: Model building and interaction with solvers and environment.

Solver: Solve an instance (instantiation of a model with data) using mathematical optimization.

- Major commercial and academic **solvers integrated**:
  31 solvers, half of them actively developed/updated
- Average number of commercial **solvers per license**:
  - Academic clients: 2.9
  - Commercial clients: 2.2
- **Switch between solvers** with one statement:
  `option solver = scip;` (since GAMS 24.5)

Solvers ↔ Problemtypes (GAMS 24.5)

GAMS

	LP	MIP	NLP	MCP	MPEC	CNS	DNLP	MINLP	QCP	MIQCP	Stoch.	Global
ALPHAECP								x		x		
ANTIGONE 1.1			x			x	x	x	x	x		x
BARON 15.8	x	x	x			x	x	x	x	x		x
BDMLP	x	x										
BONMIN 1.8								x		x		
CBC 2.9	x	x										
CONOPT 3	x		x			x	x		x			
COUENNE 0.5			x			x	x	x	x	x		x
CPLEX 12.6	x	x							x	x		
DECIS	x										x	
DICOPT								x		x		
GUROBI 6.0	x	x							x	x		
IPOPT 3.12	x		x			x	x		x			
KNITRO 9.1	x		x			x	x	x	x	x		
LGO	x		x				x		x			(x)
LINDO 9.0	x	x	x				x	x	x	x	x	x
LOCALSOLVER 5.5	x	x	x			x	x	x	x	x		
MILES				x								
MINOS	x		x			x	x		x			
MOSEK 7	x	x	x				x		x	x		
MSNLP			x				x		x			(x)
NLPEC				x	x							
OQNLP			x				x	x	x	x		(x)
PATH				x		x						
SBB								x		x		
SCIP 3.2		x	x			x	x	x	x	x		x
SNOPT	x		x			x	x		x			
SOPLEX 2.2	x											
SULUM 4.3	x	x										
XA	x	x										
XPRESS 28.01	x	x							x	x		

Independence of Model and Platform

Supported Platforms:

Solver/Platform availability - 24.5	x86 32bit MS Windows	x86 64bit MS Windows	x86 64bit Linux	x86 64bit MacOS X	x86 64bit SOLARIS	Sparc 64bit SOLARIS	IBM Power 64bit AIX
ALPHAECP	✔	✔	✔	✔	✔	✔	✔
ANTIGONE 1.1	✔	✔	✔	✔			
BARON 15.8	✔	✔	✔	✔			
BDMLP	✔	✔	✔	✔	✔	✔	✔
BONMIN 1.8	✔	✔	✔	✔	✔		
CBC 2.9	✔	✔	✔	✔	✔		
CONOPT 3	✔	✔	✔	✔	✔	✔	✔
COUENNE 0.5	✔	✔	✔	✔	✔		
CPLEX 12.6	✔	✔	✔	✔	✔	✔	✔
DECIS	✔	✔	✔			✔	
DICOPT	✔	✔	✔	✔	✔	✔	✔
GLOMIQO 2.3	✔	✔	✔	✔			
GUROBI 6.0	✔	✔	✔	✔			✔
GUSS	✔	✔	✔	✔	✔	✔	✔
IPOPT 3.12	✔	✔	✔	✔	✔		
KESTREL	✔	✔	✔	✔	✔	✔	✔
KNITRO 9.1	✔	✔	✔	✔			
LGO	✔	✔	✔	✔	✔	✔	
LINDO 9.0	✔	✔	✔	✔			
LINDOGLOBAL 9.0	✔	✔	✔	✔			
LOCALSOLVER 5.5	✔	✔	✔	✔			
MILES	✔	✔	✔	✔	✔	✔	✔
MINOS	✔	✔	✔	✔	✔	✔	✔
MOSEK 7	✔	✔	✔	✔			
MSNLP	✔	✔	✔	✔		✔	
NLPEC	✔	✔	✔	✔	✔	✔	✔
OQNLP	✔	32bit					
PATH	✔	✔	✔	✔	✔	✔	✔
SBB	✔	✔	✔	✔	✔	✔	✔
SCIP 3.2	✔	✔	✔	✔	✔		
SNOPT	✔	✔	✔	✔	✔	✔	✔
SOPLEX 2.2	✔	✔	✔	✔	✔		
SULUM 4.3	✔	✔	✔				
XA	✔	✔	✔				
XPRESS 28.01	✔	✔	✔	✔	✔	✔	✔

Tools/Platform availability - 24.5	x86 32bit MS Windows	x86 64bit MS Windows	x86 64bit Linux	x86 64bit MacOS X	x86 64bit SOLARIS	Sparc 64bit SOLARIS	IBM Power 64bit AIX
ASK	✔	32bit					
BIB2GMS	✔	✔	✔	✔	✔	✔	✔
CHK4UPD	✔	✔	✔	✔	✔	✔	✔
CHOLESKY	✔	✔	✔	✔	✔	✔	✔
CSDP	✔	✔	✔	✔	✔		
CSV2GDX	✔	✔	✔	✔	✔	✔	✔
EIGENVALUE	✔	✔	✔	✔	✔	✔	✔
EIGENVECTOR	✔	✔	✔	✔	✔	✔	✔
ENDECRYPT	✔	✔	✔	✔	✔	✔	✔
GAMSIDE	✔	32bit					
GAMS POSIX Utilities1)	✔	✔	✔	✔	✔	✔	✔
GDX2ACCESS	✔	32bit					
GDX2HAR	✔	32bit					
GDX2SQLITE	✔	✔	✔	✔	✔	✔	✔
GDX2VEDA	✔	✔	✔	✔	✔	✔	✔
GDX2XLS	✔	32bit					
GDXCOPY	✔	✔	✔	✔	✔	✔	✔
GDXDIFF	✔	✔	✔	✔	✔	✔	✔
GDXDUMP	✔	✔	✔	✔	✔	✔	✔
GDXMERGE	✔	✔	✔	✔	✔	✔	✔
GDXMRW	✔	✔	✔	✔			
GDXRANK	✔	✔	✔	✔	✔	✔	✔
GDXRENAME	✔	✔	✔	✔	✔	✔	✔
GDXRRW	✔	✔	✔	✔	src only	src only	src only
GDXTROLL	✔	✔	✔	✔	✔	✔	✔
GDXVIEWER	✔	32bit					
GDXXRW	✔	32bit					
GMSUNZIP	✔	✔	✔	✔	✔	✔	✔
HAR2GDX	✔	32bit					
IDECMDS	✔	32bit					
INVERT	✔	✔	✔	✔	✔	✔	✔
MCFILTER	✔	✔	✔	✔	✔	✔	✔
MDB2GMS	✔	32bit					
MODEL2TEX	✔	✔	✔	✔	✔	✔	✔
MPS2GMS	✔	✔	✔	✔	✔	✔	✔
MSAPPAVAIL	✔	32bit					
SCENRED	✔	✔	✔	✔	✔	✔	✔
SCENRED2	✔	✔	✔	✔	✔	✔	✔
SHELLEXECUTE	✔	32bit					
SQL2GMS	✔	32bit					

Documentation and Help

Online: `http://www.gams.com/help` (with search)

Offline: `<GAMS system directory>/docs/index.html` (no search, use grep!)

- **GAMS – A User's Guide**: Tutorial, Basics, Advanced Topics
- **McCarl (Expanded) GAMS User Guide**
- **Solver Manuals**
- **Tools Manuals**
- **APIs**: Tutorials and Reference Manuals
- **Release Notes**

Tutorial Videos: `http://www.youtube.com/user/GAMSLessons`

Support wiki: `http://support.gams.com/doku.php`

Discussion group: `http://www.gamsworld.org/`

Model Libraries

Online: `http://www.gams.com/modlibs`

Offline: `gamslib`, `apilib`, `datalib`, `emplib`, `testlib` tools

- **GAMS Model Library**
  - representing interesting and sometimes classic problems
  - illustrating GAMS modeling capabilities

Model Libraries

GAMS

Online: `http://www.gams.com/modlibs`

Offline: `gamslib`, `apilib`, `datalib`, `emplib`, `testlib` tools

- **GAMS Model Library**
  - representing interesting and sometimes classic problems
  - illustrating GAMS modeling capabilities
- **GAMS API Library**
  - scripts to compile and execute GAMS API examples
- **GAMS Data Utilities Library**
  - demonstrate utilities to interface GAMS with other applications
- **GAMS EMP Library**
  - illustrate and test capabilities of extended mathematical programming facility
- **Contributed Libraries**:
  - FINLIB – financial optimization models (by Consiglio, Nielsen and Zenios)
  - NOALIB – nonlinear optimization applications models (by Neculai Andrei)
- **GAMS Testlib Library**
  - testing and quality control

Recall Tuesday morning . . .

The Equitable Coach Problem

What is important?

- How many players are in the team? How many on the field?
- What is the formation that the team will play? 2-3-1, 2-2-2 or 3-1-2?
- How many interchanges do you want to perform?
- What are the positional preferences of the players?
- Are you equitable for only a single game or across the two matches?
- Complicated: Does winning the first match impact your decision about being equitable for the second?

Burt, Maher, Witzig – Modelling 11 / 102

GLEBE GORILLAS

Problem definition

OBJECTIVE: For a single game identify the equitable number of playing minutes for each player given that there is only one goal keeper and players have positional preferences.

- 7 players on the field, 12 players in a team
- 20 minutes games ⇒ 140 player minutes
- 4 different position types, Goal Keeper, Defence, Midfield and Striker.

Solution is not as easy. Need a more sophisticated mathematical model.

Burt, Maher, Witzig – Modelling 12 / 102

The Equitable Coach Problem: disaggregated model

We now consider the version of the ECP where we allocate players to positions at particular times.

- $x_{r,i,t}$ [binary] is 1 if player r plays in position i in play period t;
- ϵ_r^+ [continuous] is the number of periods player r plays above the average;
- ϵ_r^- [continuous] is the number of periods player r plays below the average.
- R is the set of players;
- N is the set of positions;
- T is the total number of game minutes.

Burt, Maher, Witzig – Modelling 88 / 102

The Equitable Coach Problem: disaggregated model

$$
\begin{aligned}
ECPd: \quad & \min \sum_r (\epsilon_r^+ + \epsilon_r^-) & & \\
& \sum_r x_{r,i,t} = 1 & \forall\, t \in T, i \in N, & \quad (8)\\
& \sum_i x_{r,i,t} \le 1 & \forall\, t \in T, r \in R, & \quad (9)\\
& \sum_{t,i} x_{r,i,t} - \sum_{t,i,r'} \frac{x_{r',i,t}}{|R|} = \epsilon_r^+ - \epsilon_r^- & \forall\, r \in R, & \quad (10)\\
& \boxed{x_{r,i,t} \in [0,1]} & \forall r \in R,\ i \in N,\ t \in T, & \\
& \epsilon_r^+, \epsilon_r^- \in \mathbb{R}_{\ge 0}. & &
\end{aligned}
$$

Burt, Maher, Witzig – Modelling 89 / 102

Recall: Equitable Coach Problem (ECP)

GAMS

- 7 players on the field, 12 players in a team
- 20 minutes games $\Rightarrow$ 140 player minutes
- 4 different position types, Goal Keeper, Defence, Midfield and Striker.

OBJECTIVE: For a single game identify the **positions of players at particular times** given that players have positional preferences and such that **playing times are as equitable as possible**.

Recall: Equitable Coach Problem (ECP)

GAMS

- 7 players on the field, 12 players in a team
- 20 minutes games ⇒ 140 player minutes
- 4 different position types, Goal Keeper, Defence, Midfield and Striker.

OBJECTIVE: For a single game identify the **positions of players at particular times** given that players have positional preferences and such that **playing times are as equitable as possible**.

$x_{r,i,t}$ is 1 if player r plays in position i in play period t;

$\epsilon_r^{+/-}$ is the number of periods player r plays above/below the average;

R is the set of players;

R_i is the set of players than can play on position i;

P is the set of positions;

T is the total number of game minutes.

$$\min \sum_r \left(\epsilon_r^+ + \epsilon_r^-\right)$$

$$\sum_{r \in R_i} x_{r,i,t} = 1 \qquad \forall\, t \in T, i \in P,$$

$$\sum_{i:r \in R_i} x_{r,i,t} \leq 1 \qquad \forall\, t \in T, r \in R,$$

$$\sum_{t,i:r \in R_i} x_{r,i,t} - \sum_{t,i,r' \in R_i} \frac{x_{r',i,t}}{|R|} = \epsilon_r^+ - \epsilon_r^- \quad \forall r \in R,$$

$$x_{r,i,t} \in [0,1] \qquad \forall r \in R,\ i \in N,\ t \in T,$$

$$\epsilon_r^+, \epsilon_r^- \in \mathbb{R}_{\geq 0}.$$

ECPd.zpl (simplified version)

GAMS

```
param gametime      := 20;
param nsubperiods   := 6;

# formation
param nkeeper       := 1;
param ndefender     := 2;
param nmidfielder   := 3;
param nstriker      := 1;

# set number of positions
param npositions    := nkeeper + ndefender + nmidfielder + nstriker;

# sets of players, positions, and minutes per game
set Player          := {
  "Josh",   "Simon", "Jordy", "Chris",  "Andy",  "Richie",
  "Tritto", "Guy",   "Neil",  "Justin", "Steve", "Cory"
};
set Classes     := { "K", "D", "M", "S" };
set Positions   := { 1 .. npositions };
set Periods     := { 1 .. nsubperiods };

...

var x[Player * Positions * Periods] binary;
var eplus[Player] real;
var eminus[Player] real;

# the model:
#  1.) objective function
#  2.) ensure that each position is used at each point in time
#  3.) ensure that each player plays on at most one position per time
#  4.) calculate the minutes over/below the average
minimize cost: sum <r> in Player : gametime/nsubperiods * (eplus[r] + eminus[r]);

subto c1: forall <i> in Positions :
              forall <t> in Periods : sum <r> in Player with pos[r,GetClass(i)] == 1: x[r,i,t] == 1;

subto c2: forall <r> in Player :
              forall <t> in Periods : sum <i> in Positions with pos[r,GetClass(i)] == 1: x[r,i,t] <= 1;
...
```

ECPd.gms

GAMS

```
$title Equitable Coach Problem

* timing parameters
* param gametime       := 20;
scalar gametime / 20 /;
* param nsubperiods    := 6;
$set nsubperiods 6

* formation 1—2—3—1
* param nkeeper        := 1;
* param ndefender      := 2;
* param nmidfielder    := 3;
* param nstriker       := 1;
scalar nkeeper / 1 /;
scalar ndefender / 2 /;
scalar nmidfielder / 3 /;
scalar nstriker / 1 /;

* set number of positions
* param npositions     := nkeeper + ndefender + nmidfielder + nstriker;
$set npositions 7
```

ECPd.gms

GAMS

```
* sets of players, positions, and minutes per game
*set Player          := {
*  "Josh",   "Simon", "Jordy", "Chris",  "Andy",  "Richie",
*  "Tritto", "Guy",   "Neil",  "Justin", "Steve", "Cory"
*};
set r "Player" /
  "Josh",   "Simon", "Jordy", "Chris",  "Andy",  "Richie",
  "Tritto", "Guy",   "Neil",  "Justin", "Steve", "Cory"
/;

*set Classes    := { "K", "D", "M", "S" };
*set Positions  := { 1 .. npositions };
*set Periods    := { 1 .. nsubperiods };
set c "Position classes" / K, D, M, S /;
set p "Positions" / 1 * %npositions% /;
set t "Periods" / 1 * %nsubperiods% /;
```

ECPd.gms

```
* entry (i,j) is 1 <=> player i can play on position j
$ontext
param pos[Player * Classes] := | "K", "D", "M", "S" |
                      |"Josh"  |   0,   0,   1,   1  |
                      |"Simon" |   0,   0,   0,   1  |
...
                      |"Steve" |   0,   1,   0,   0  |
                      |"Cory"  |   1,   0,   0,   0  |;
$offtext
Table pos(r,c)
         K  D  M  S
 Josh          1  1
 Simon            1
 Jordy            1
 Chris      1  1
 Andy          1
 Richie     1  1
 Tritto     1
 Guy        1
 Neil          1
 Justin        1
 Steve      1
 Cory    1
;
```

ECPd.gms

```
$ontext
defstrg GetClass(i) := if i <= nkeeper then "K"
                       else
                          if i <= nkeeper + ndefender then "D"
                          else
                             if i <= nkeeper + ndefender + nmidfielder
                             else
                                "S"
                             end
                          end
                       end;
$offtext
set pos2class(p,c);
loop(p,
  if( p.val <= nkeeper,
    pos2class(p,'K') = yes;
  else if( p.val <= nkeeper + ndefender,
    pos2class(p,'D') = yes;
  else if( p.val <= nkeeper + ndefender + nmidfielder,
    pos2class(p,'M') = yes;
  else
    pos2class(p,'S') = yes;
  )))
);
```

```
set canplay(r,p) "whether player r can play in position p";
* r can play at p if pos(r,c) is 1 for the position class that p belongs t
canplay(r,p) = (sum(c$pos2class(p,c), pos(r,c)) = 1);

display pos, pos2class, canplay;


* variable definition
*var x[Player * Positions * Periods] binary;
*var eplus[Player] real;
*var eminus[Player] real;
Binary Variable x(r,p,t);
Positive Variables eplus(r), eminus(r);

* the model:
*  1.) objective function
*  2.) ensure that each position is used at each point in time
*  3.) ensure that each player plays on at most one position per time
*  4.) calculate the minutes over/below the average
Variable inquity;
Equations obj, c1, c2, c3;
```

ECPd.gms

```
* minimize cost: sum <r> in Player : gametime/nsubperiods * (eplus[r]+emin
obj.. inquity =E= sum(r, gametime/%nsubperiods% * (eplus(r) + eminus(r)));

*subto c1: forall <i> in Positions : forall <t> in Periods :
*       sum <r> in Player with pos[r,GetClass(i)] == 1: x[r,i,t] == 1;
c1(p,t).. sum(r$canplay(r,p), x(r,p,t)) =E= 1;

*subto c2: forall <r> in Player : forall <t> in Periods :
*       sum <i> in Positions with pos[r,GetClass(i)] == 1: x[r,i,t] <= 1;
c2(r,t).. sum(canplay(r,p), x(r,p,t)) =L= 1;

*subto c3: forall <r> in Player :
*   sum <i,t> in Positions * Periods with pos[r,GetClass(i)] == 1 : x[r,i,t
* -sum <q,i,t> in Player * Positions * Periods with
*    pos[q,GetClass(i)] == 1 : (x[q,i,t]/card(Player)) == eplus[r]-eminus[r
alias(r,q);
c3(r).. sum((p,t)$canplay(r,p), x(r,p,t))
      - sum((q,p,t)$canplay(q,p), x(q,p,t)/card(r)) =E= eplus(r)-eminus(r)

* declare model ECPd, consisting of all equations seen so far
Model ECPd / all /;

* solve ECPd (minimizing inquity) as MIP to optimality (gap tolerance 0)
option optcr = 0;
Solve ECPd minimizing inquity using MIP;
```

Exercise

GAMS

TASK: Extend the equitable coach problem model by restricting the number of position changes for all players to 3.

- Get `http://co-at-work.zib.de/files/gams/ECP_exercise.gms`
- Open the file in your favorite text editor and read the more detailed instructions at the beginning.
- Run the model: `gams ECP_exercise.gms errmsg=1`
- In case of compilation or execution errors, e.g.,

```
--- Starting compilation
--- ECP_exercise.gms(111) 3 Mb 3 Errors
*** Status: Compilation error(s)
--- Job ECP_exercise.gms Stop 10/01/15 20:15:47 elapsed 0:00
```

  see listing file (`ECP_exercise.lst`):

```
  90  c2(r,t).. sum(canplay(r,p), x(r,p,t)) <= 1;
****                                           $37
****  37  '=l=' or '=e=' or '=g=' operator expected
```

Independence of Model and Data

GDX – **G**AMS **D**ata E**x**change:

- **Binary data** format for fast exchange of data with GAMS
- Stores **sets**, **parameters**, **values of variables/equations** with domain information, but no symbolic information (e.g., equation algebra)
- **Consistency**: no duplicates or contradictions
- **Platform independent**
- Can be **compressed**

Independence of Model and Data

GDX – **G**AMS **D**ata **Ex**change:

- **Binary data** format for fast exchange of data with GAMS
- Stores **sets**, **parameters**, **values of variables/equations** with domain information, but no symbolic information (e.g., equation algebra)
- **Consistency**: no duplicates or contradictions
- **Platform independent**
- Can be **compressed**
- **Read and write** in GAMS:
  - on command line: parameter `gdx=<filename>`
  - during compilation: `$gdxin`, `$gdxout`, `$load`, `$unload`, ...
  - during execution: `execute_load`, `execute_unload`, ...
- **Tools and APIs** to read and write from other environments:
  - `gdxdump`, `csv2gdx`, `gdxviewer` (win only), ...
  - Matlab, MS Access, MS Excel, ODBC/SQL, R, SQLite
  - low-level APIs for C, C++, C#, Delphi, Fortran, Java, Python, VBA, VB.NET
  - high-level APIs for C#, Java, and Python

High-level APIs (C#, Java, Python) to embed GAMS

Creating input:

- Read GAMS model from file or string (APIs have no modeling capability!)
- Create input data (wrapper around GDX API)

Callout to GAMS:

- Set GAMS options (e.g., solver to use, gap limit, ...)
- Start GAMS run and store results in GDX file

Processing results:

- Read and process result data (wrapper around GDX API)

High-level APIs (C#, Java, Python) to embed GAMS

GAMS

Creating input:

- Read GAMS model from file or string (APIs have no modeling capability!)
- Create input data (wrapper around GDX API)

Callout to GAMS:

- Set GAMS options (e.g., solver to use, gap limit, ...)
- Start GAMS run and store results in GDX file

Processing results:

- Read and process result data (wrapper around GDX API)

Main Classes:

- `GAMSWorkspace`: Environment
- `GAMSJob`: Storing and running a model
- `GAMSDatabase`: In-memory representation of data ("GDX in-memory")
- `GAMSOptions`: Manipulation of GAMS options
- `GAMSModelInstance`: Hot-start solve for a sequence of closely related problems (changes in numerical values of matrix, objective, sides, bounds only)

Example (Python)

<GAMS system directory>/apifiles/Python/transport1.py:

```python
from gams import *
import sys

if len(sys.argv) > 1:
    ws = GamsWorkspace(system_directory = sys.argv[1])
else:
    ws = GamsWorkspace()

ws.gamslib("trnsport")    # get 'trnsport' from GAMS model library
t1 = ws.add_job_from_file("trnsport.gms")

opt = ws.add_options()
opt.all_model_types = "soplex"
t1.run(opt)

for rec in t1.out_db["x"]:
    print("x(" + rec.key(0) + "," + rec.key(1) + "): " + \
          "level=" + str(rec.level) + " marginal=" + str(rec.marginal))
```

```
x(seattle,new-york): level=50.0 marginal=0.0
x(seattle,chicago): level=300.0 marginal=0.0
x(seattle,topeka): level=0.0 marginal=0.036000000000000004
x(san-diego,new-york): level=275.0 marginal=0.0
x(san-diego,chicago): level=0.0 marginal=0.009000000000000008
x(san-diego,topeka): level=275.0 marginal=0.0
```

Cutting Stock by Column Generation

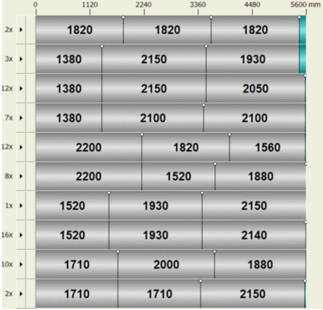
Problem: [Gilmore, Gomory, 1961, 1963]

- Cut out some paper products of different sizes from large raw paper rolls, in order to meet a customer's order.
- The objective is to minimize the required number of the paper rolls.

Example:

- Raw paper roll width: 5600mm
- Customer's Order:

Width	Rolls	Width	Rolls
1380	22	2000	10
1520	25	2050	12
1560	12	2100	14
1710	14	2140	16
1820	18	2150	18
1880	18	2200	20
1930	20		



Source: Wikipedia

Cutting Stock Problem Formulation

GAMS

Width of raw paper roll: W

Customer order:

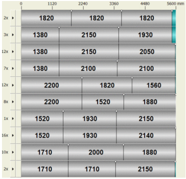
- Set I of widths.
- Widths $w_i \in \mathbb{N}$, $i \in I$.
- Required number of rolls for each width $d_i \in \mathbb{N}$, $i \in I$.

$W = 5600$

Order:

	Width	Rolls
I	w_i	d_i
1	1380	22
2	1520	25
3	1560	12
4	1710	14
5	1820	18
6	1880	18
7	1930	20
8	2000	10
9	2050	12
10	2100	14
11	2140	16
12	2150	18
13	2200	20

Solution:



Cutting Stock Problem Formulation

GAMS

Width of raw paper roll: W

Customer order:

- Set I of widths.
- Widths $w_i \in \mathbb{N}$, $i \in I$.
- Required number of rolls for each width $d_i \in \mathbb{N}$, $i \in I$.

Set of valid patterns P:

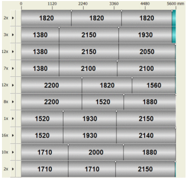
- $a_{p,i} \in \mathbb{N}$, $p \in P$, $i \in I$: Number of width i in pattern p
- e.g., $a_{p,\cdot} = (1,0,0,0,0,0,0,0,0,2,0,0,0) \Leftrightarrow 1 \times 1380, 2 \times 2100$
- pattern only valid if $\sum_{i \in I} a_{p,i} w_i \leq W$.

$W = 5600$

Order:

	Width	Rolls
I	w_i	d_i
1	1380	22
2	1520	25
3	1560	12
4	1710	14
5	1820	18
6	1880	18
7	1930	20
8	2000	10
9	2050	12
10	2100	14
11	2140	16
12	2150	18
13	2200	20

Solution:



Cutting Stock Problem Formulation

Width of raw paper roll: W

Customer order:

- Set I of widths.
- Widths $w_i \in \mathbb{N}$, $i \in I$.
- Required number of rolls for each width $d_i \in \mathbb{N}$, $i \in I$.

Set of valid patterns P:

- $a_{p,i} \in \mathbb{N}$, $p \in P$, $i \in I$: Number of width i in pattern p
- e.g., $a_{p,\cdot} = (1,0,0,0,0,0,0,0,0,2,0,0,0) \Leftrightarrow 1 \times 1380, 2 \times 2100$
- pattern only valid if $\sum_{i \in I} a_{p,i} w_i \leq W$.

Model:

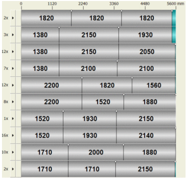
- Variable $x_p \in \mathbb{N}$: how often to use pattern p
- Satisfy demand: $\sum_{p \in P} a_{p,i}\, x_p \geq d_i$ for all $i \in I$
- Minimize number of patterns: $\min \sum_{p \in P} x_p$

$W = 5600$

Order:

I	Width w_i	Rolls d_i
1	1380	22
2	1520	25
3	1560	12
4	1710	14
5	1820	18
6	1880	18
7	1930	20
8	2000	10
9	2050	12
10	2100	14
11	2140	16
12	2150	18
13	2200	20

Solution:



Cutting Stock Problem Formulation

Width of raw paper roll: W

Customer order:

- Set I of widths.
- Widths $w_i \in \mathbb{N}$, $i \in I$.
- Required number of rolls for each width $d_i \in \mathbb{N}$, $i \in I$.

Set of valid patterns P:

- $a_{p,i} \in \mathbb{N}$, $p \in P$, $i \in I$: Number of width i in pattern p
- e.g., $a_{p,\cdot} = (1,0,0,0,0,0,0,0,0,2,0,0,0) \Leftrightarrow 1 \times 1380, 2 \times 2100$
- pattern only valid if $\sum_{i \in I} a_{p,i} w_i \leq W$.

Model:

- Variable $x_p \in \mathbb{N}$: how often to use pattern p
- Satisfy demand: $\sum_{p \in P} a_{p,i}\, x_p \geq d_i$ for all $i \in I$
- Minimize number of patterns: $\min \sum_{p \in P} x_p$

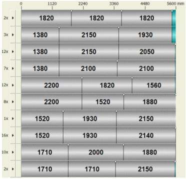
Observation: Only a few of all valid patterns will be used in a solution $\Rightarrow$ **generate set of active patterns** dynamically

$W = 5600$

Order:

I	Width w_i	Rolls d_i
1	1380	22
2	1520	25
3	1560	12
4	1710	14
5	1820	18
6	1880	18
7	1930	20
8	2000	10
9	2050	12
10	2100	14
11	2140	16
12	2150	18
13	2200	20

Solution:



Restricted Master and Pricing Problems

Let $\mathcal{P} \subseteq P$ be the **set of generated pattern**.
Restricted Master Problem:

$$\begin{aligned} \min \ & \sum_{p \in \mathcal{P}} x_p \\ \text{s.t.} \ & \sum_{p \in \mathcal{P}} a_{p,i}\, x_p \geq d_i && \forall i \in I \\ & x_p \in \mathbb{R}_+ && \forall p \in \mathcal{P} \end{aligned} \tag{RMP}$$

Let μ be the dual variable associated with the inequality constraint in (RMP).

Pricing Problem:

$$\begin{aligned} \min \ & 1 - \sum_{i \in I} \mu_i y_i \\ \text{s.t.} \ & \sum_{i \in I} w_i y_i \leq W \\ & y_i \in \mathbb{N}_+ && \forall i \in I \end{aligned} \tag{PP}$$

If (PP) has a solution y^* with value < 0, an improving pattern has been found $\Rightarrow$ add to $\mathcal{P}$ with $a_{p,i} = y_i^*$.

Restricted Master Problem in GAMS

$$\min \left\{ \sum_{p\in\mathcal{P}} x_p \;:\; \sum_{p\in\mathcal{P}} a_{p,i}\, x_p \geq d_i \;\forall i \in I, \quad x_p \in \mathbb{R}_+ \right\} \qquad \text{(RMP)}$$

```
Set       i     "widths"     / w1*w4 /;
Parameter r     "raw width" / 100 /
          w(i) "width"      / w1 45, w2  36, w3  31, w4  14 /
          d(i) "demand"     / w1 97, w2 610, w3 395, w4 211 /;
Set   p         "possible patterns"  / 1*1000 /
      pp(p)     "dynamic subset of p";
Parameter
      api(p,i) "number of width i in pattern p";

Variable xp(p)  "how often to use pattern p"
         z      "objective variable"
Integer Variable xp;
xp.up(p) = sum(i, d(i));

Equations numpat    "number of patterns used"
          demand(i) "meet demand";
numpat..     z =e= sum(pp, xp(pp));
demand(i)..  sum(pp, api(pp,i)*xp(pp)) =g= d(i);

Model master / numpat, demand /;
Solve master min z using RMIP;
```

Pricing Problem in GAMS

$$\min \left\{ 1 - \sum_{i \in I} \mu_i y_i \;:\; \sum_{i \in I} w_i y_i \leq W, \quad y_i \in \mathbb{N}_+ \right\} \tag{PP}$$

```
Set        i    "widths"     / w1*w4 /;
Parameter  r    "raw width"  / 100 /
           w(i) "width"      / w1 45, w2 36, w3 31, w4 14 /;

Variable   z    "objective variable"
           y(i) "new pattern";
Integer variable y;
y.up(i) = ceil(r/w(i));

Equations defobj   "pricing problem objective"
          knapsack "knapsack constraint";

defobj..      z =e= 1 - sum(i, demdual(i)*y(i));
knapsack..    sum(i, w(i)*y(i)) =l= r;

Model pricing / defobj, knapsack /;
Solve pricing min z using MIP;
```

Exercise

Task: Get the Python file `cutstock.py` and add the Column-Generation loop:

1. Solve (RMP) as LP.
2. Update μ in (PP) and solve (PP).
3. If optimal value of (PP) is > 0, then construct new pattern for (RMP) from solution of (PP) and go to 1.

Final master objective value should be **452.25**.

Exercise

Task: Get the Python file `cutstock.py` and add the Column-Generation loop:

1. Solve (RMP) as LP.
2. Update μ in (PP) and solve (PP).
3. If optimal value of (PP) is > 0, then construct new pattern for (RMP) from solution of (PP) and go to 1.

Final master objective value should be **452.25**.

Solution File: `cutstock_sol.py`