Solving Difficult MIP Problems using GAMS and Condor

Michael R. Bussieck
MBussieck@gams.com

GAMS Software GmbH
http://www.gams.de

Michael C. Ferris
Ferris@cs.wisc.edu

University of Wisconsin-Madison
http://www.cs.wisc.edu/~ferris/

OR 2006
Karlsruhe, Germany, September 6-8, 2006

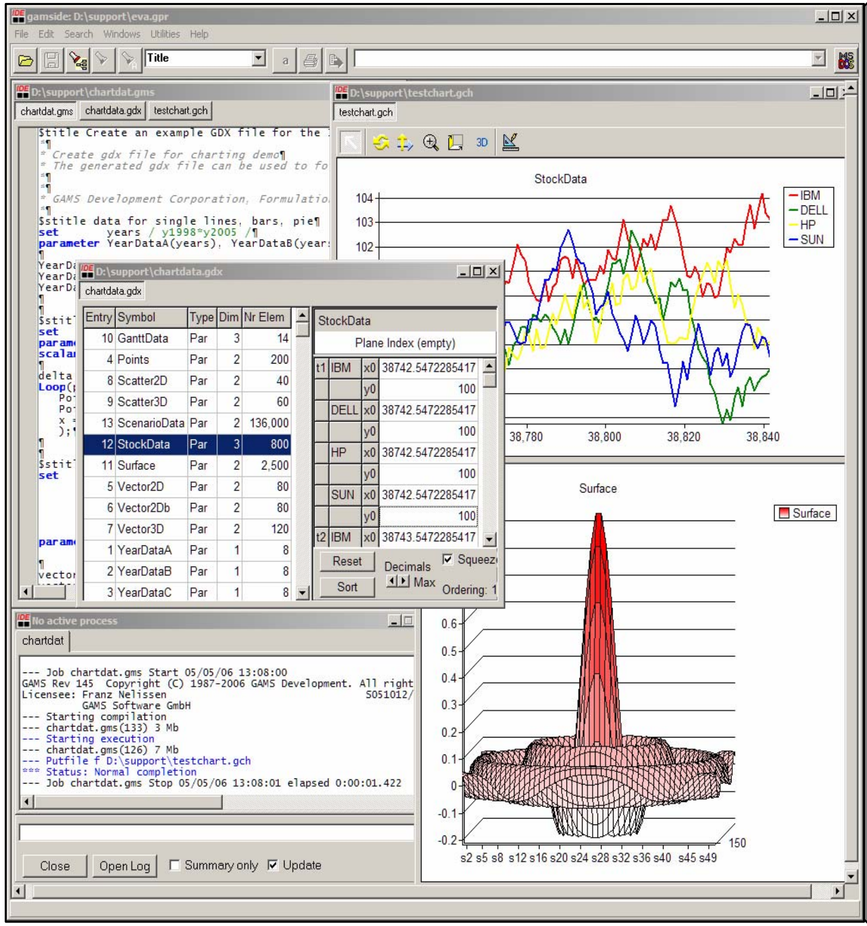
GAMS

GAMS Development / GAMS Software

- Roots: **Research project** World Bank 1976
- Pioneer in **Algebraic Modeling Systems** used for economic modeling
- Went **commercial** in 1987
- **Offices** in Washington, D.C and Cologne

- Professional **software tool provider**
- Operating in a **segmented niche market**
- Broad **academic** & **commercial** user base and network

GAMS at a Glance



General **A**lgebraic **M**odeling **S**ystem:

Algebraic Modeling Language, Integrated Solver, Model Libraries, Connectivity- & Productivity Tools

Design Principles:

- Balanced mix of declarative and procedural elements
- Open architecture and interfaces to other systems
- Different layers with separation of:
  - model and data
  - model and solution methods
  - model and operating system
  - model and interface

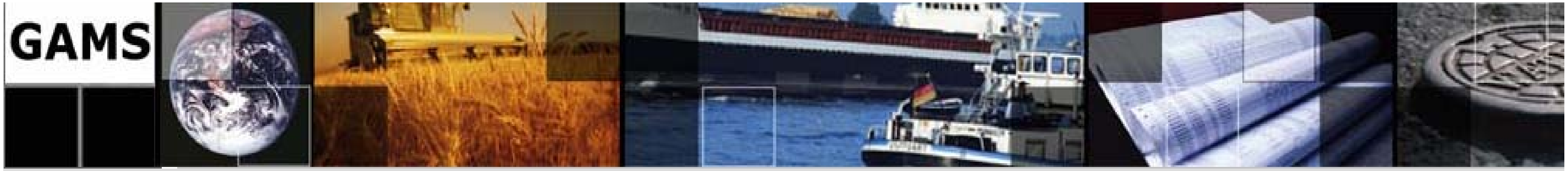
What's New???

- Improvements on all frontiers
  - Connectivity Tools
    - Databases
    - Spreadsheets
    - Specialized Visualization Tools (e.g. VEDA)
  - Productivity Tools
    - IDE Improvements
    - Charting Engine
  - Interfaces
    - Using GAMS from Application Environments
  - Solver Interfacing
    - Branch-and-Cut-and-Heuristic (BCH) Facility
    - Grid Computing

What is Grid Computing?

A pool of connected computers managed and available as a common computing resource

- Effective sharing of CPU power
- Massive parallel task execution
- Scheduler handles management tasks
- E.g. Condor, Sun N6 Grid Engine, Globus
- Can be rented or owned in common
- Licensing & security issues

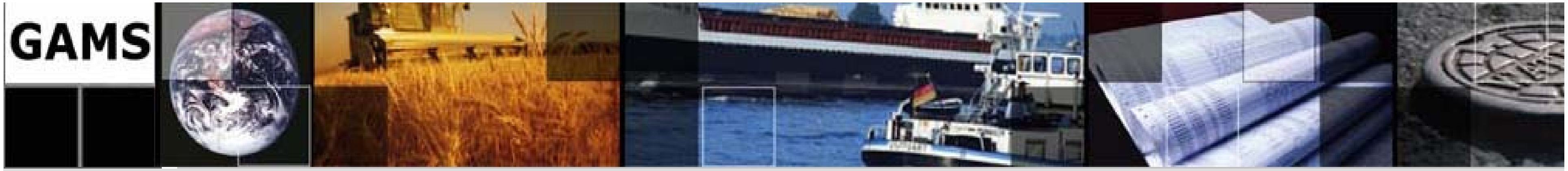


Typical Application for GAMS & Grid

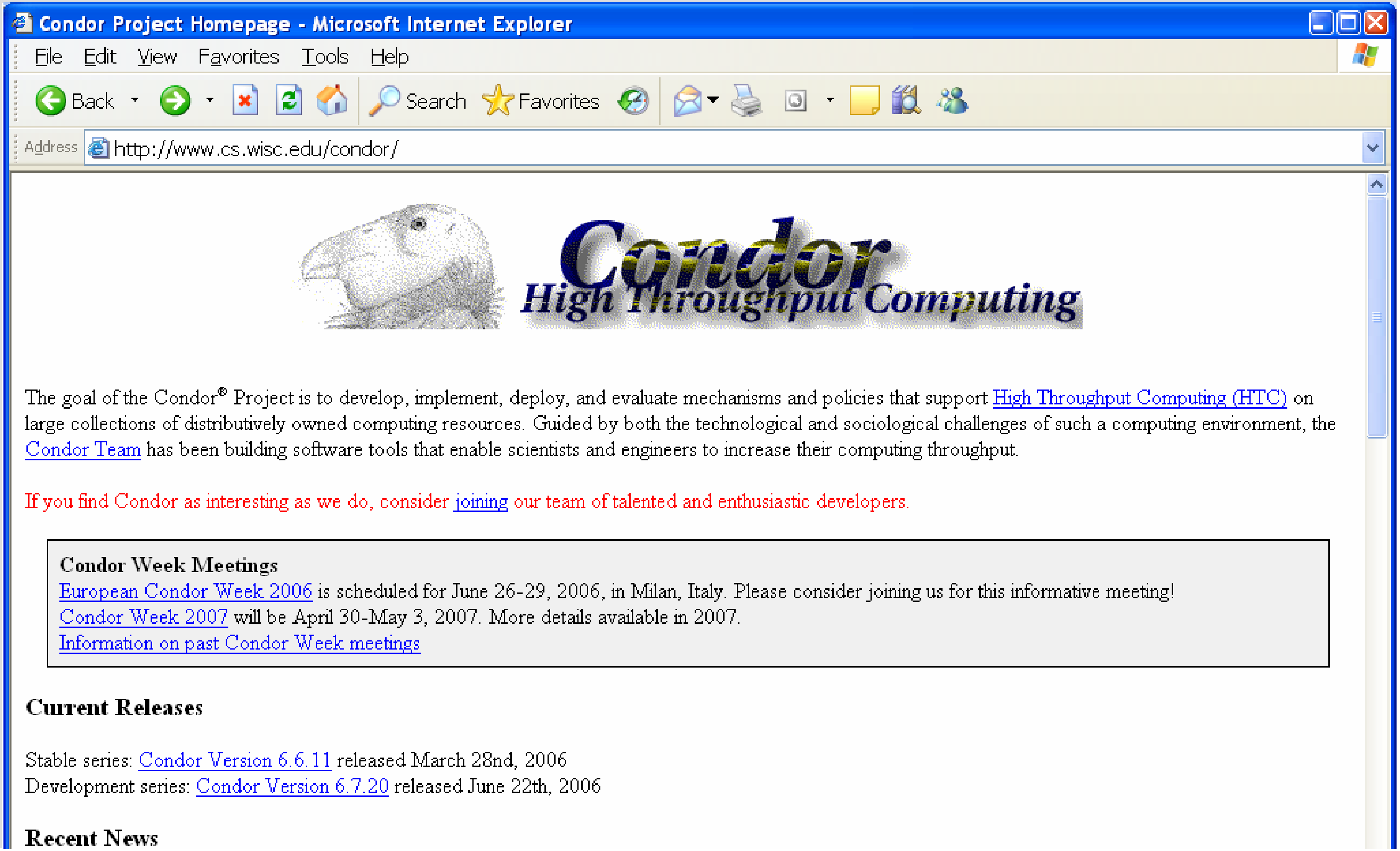
```
mymodel.solvelink=3;
loop(scenario,
  demand=sdemand(scenario); cost=scost(scenario);
  solve mymodel min obj using minlp;
  report(scenario) = var.l);

Repeat
  loop(scenario$h(scenario),
    if(handlestatus(h(scenario))=2,
      mymodel.handle=h(scenario); h(scenario)=0;
      execute_loadhandle mymodel;
      report(scenario)=var.l);
  if(card(h), execute 'sleep 1');
until card(h)=0 or timeelapsed > 100;
```

Massively Parallel MIP

- MIP/B&C Algorithm ideal to parallelize
  - Master/Worker Paradigm (process nodes in parallel)
    - Software: FATCOP/Condor, BCP/PVM, PICO/MPI
  - A-priori subdivision into *n* independent problems
    - Seymour problem solved that way
  - Open Pit Mining (openpit in GAMS Model library)
    - Partitioning integer variables to subdivide model into into 4096 sub-problems
    - Experiments (Ferris) at UW using Condor Pool



Condor



Condor Project Homepage - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Search Favorites

Address http://www.cs.wisc.edu/condor/

Condor
High Throughput Computing

The goal of the Condor® Project is to develop, implement, deploy, and evaluate mechanisms and policies that support High Throughput Computing (HTC) on large collections of distributively owned computing resources. Guided by both the technological and sociological challenges of such a computing environment, the Condor Team has been building software tools that enable scientists and engineers to increase their computing throughput.

If you find Condor as interesting as we do, consider joining our team of talented and enthusiastic developers.

Condor Week Meetings
European Condor Week 2006 is scheduled for June 26-29, 2006, in Milan, Italy. Please consider joining us for this informative meeting!
Condor Week 2007 will be April 30-May 3, 2007. More details available in 2007.
Information on past Condor Week meetings

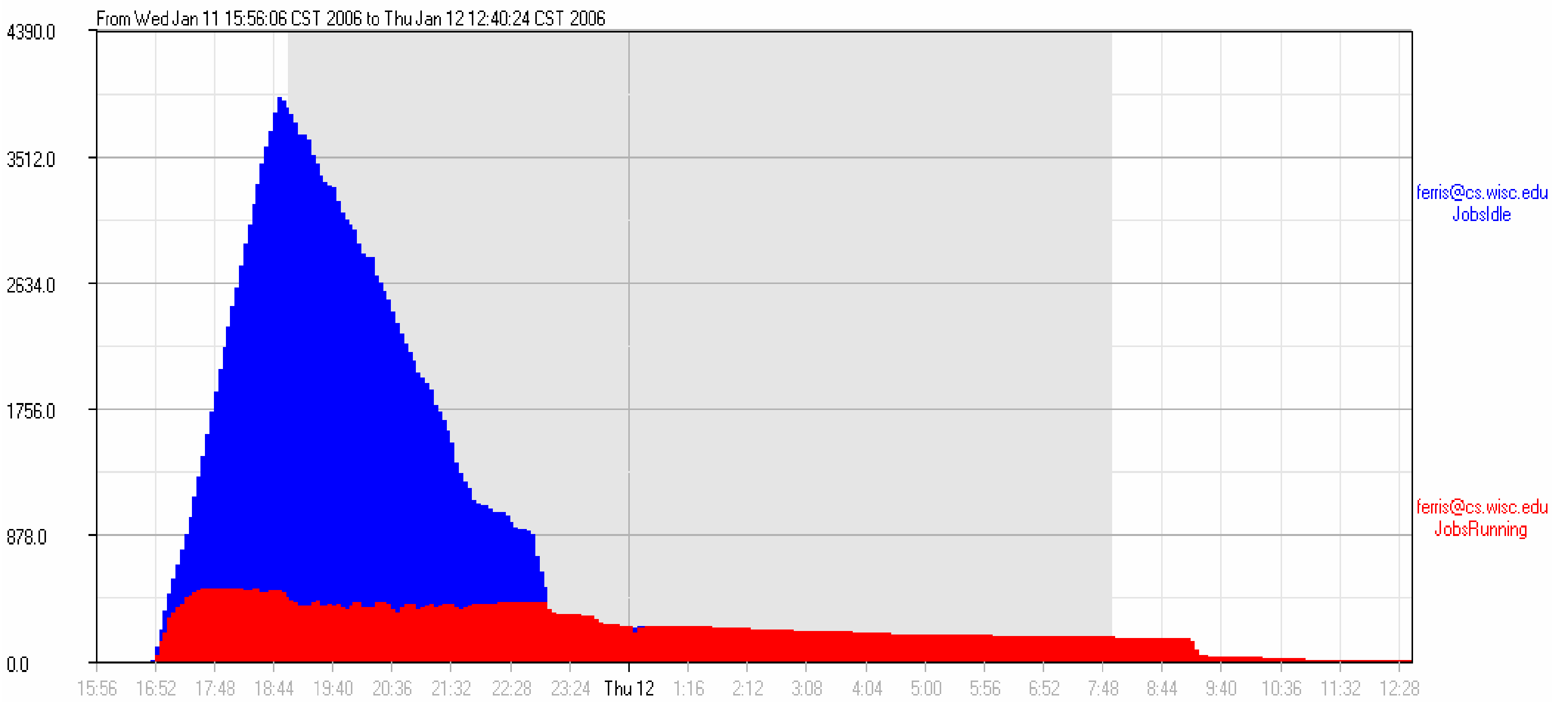
Current Releases

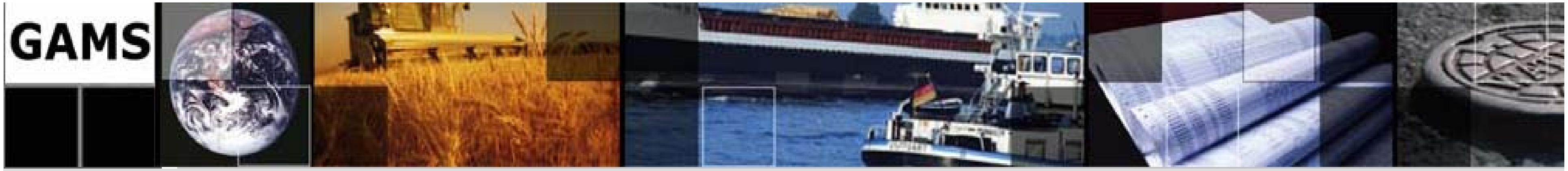
Stable series: Condor Version 6.6.11 released March 28nd, 2006
Development series: Condor Version 6.7.20 released June 22th, 2006

Recent News

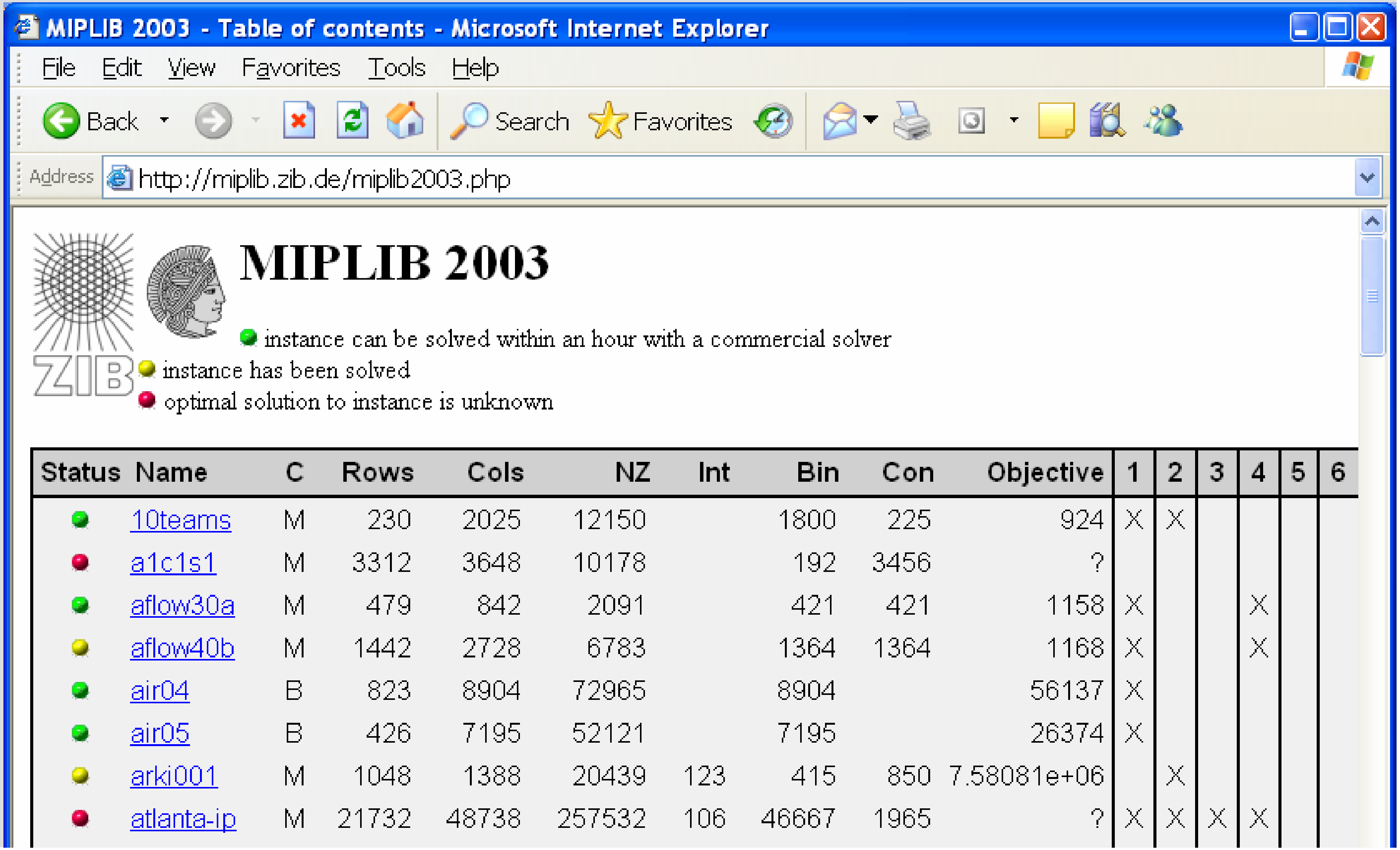
Results for 4096 MIPS on Condor Grid

- Submission started Jan 11,16:00
- All jobs submitted by Jan 11, 23:00
- All jobs returned by Jan 12, 12:40
  - 20 hours wall time, 5000 CPU hours
  - Peak number of CPU’s: 500





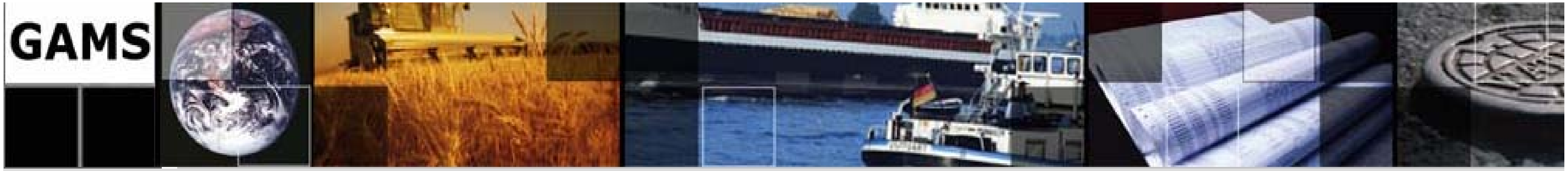
Testing MIPLIB2003 Instances



MIPLIB 2003

- instance can be solved within an hour with a commercial solver
- instance has been solved
- optimal solution to instance is unknown

Status	Name	C	Rows	Cols	NZ	Int	Bin	Con	Objective	1	2	3	4	5	6
	10teams	M	230	2025	12150		1800	225	924	X	X				
	a1c1s1	M	3312	3648	10178		192	3456	?						
	aflow30a	M	479	842	2091		421	421	1158	X			X		
	aflow40b	M	1442	2728	6783		1364	1364	1168	X			X		
	air04	B	823	8904	72965		8904		56137	X					
	air05	B	426	7195	52121		7195		26374	X					
	arki001	M	1048	1388	20439	123	415	850	7.58081e+06		X				
	atlanta-ip	M	21732	48738	257532	106	46667	1965	?	X	X	X	X		



Tool and expertise combined

- Initial schemes take over 1 year of computation and go nowhere – even with fastest commercial solver like CPLEX/XPRESS
- Extensions of approach that incorporate both computational strategies and optimization expertise
  – Adaptive refinement strategy
  – Sophisticated problem domain branching and cuts
  – Use of resources beyond local file system
  – Dedicated resources

Problems with a-priori Partitioning

- 99% of sub-problems very easy to solve
- 1% (almost) as difficult as the original problem

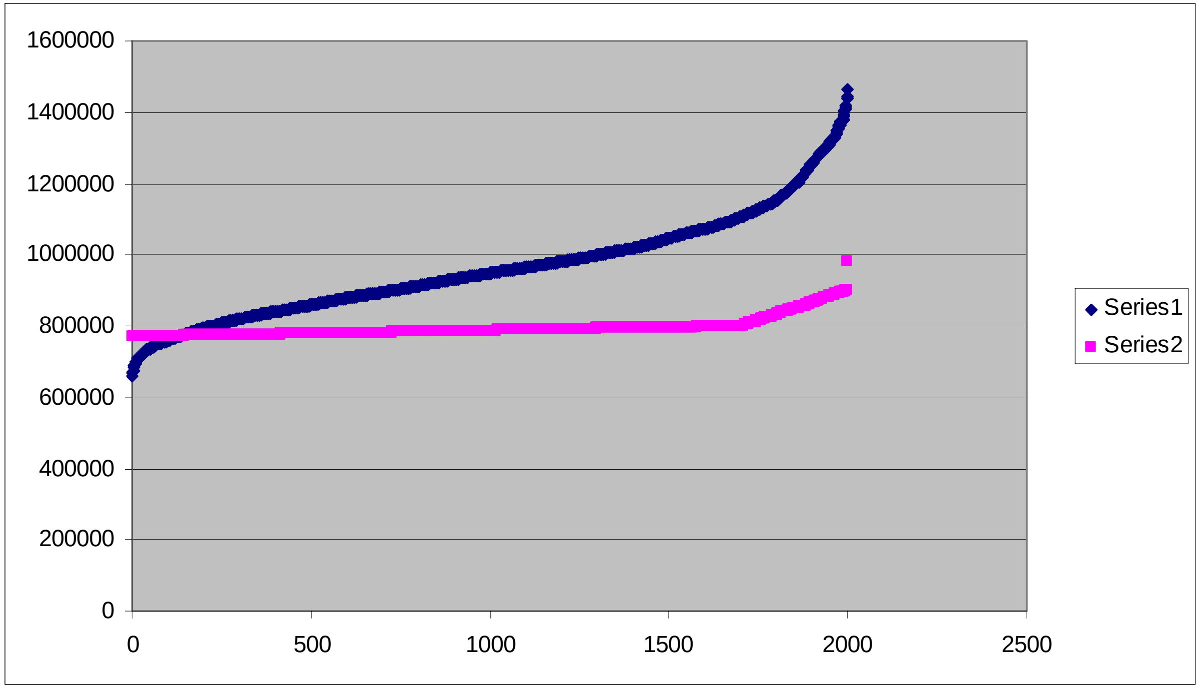
- How can we find *n* sub-problems with similar (but reduced) level of difficulty?
  – B&C Code keeps a list of *open/unexplored* nodes
  – Problem-bounds of these open nodes represent partitioning of the original problem

Node	Nodes Left	Objective	IInf	Best Integer	Cuts/ Best Node	ItCnt	Gap
0	0	29.6862	64		29.6862	165	
100	37	17.0000	14		25.0000	2230	
200	70	21.8429	22		24.0000	4022	

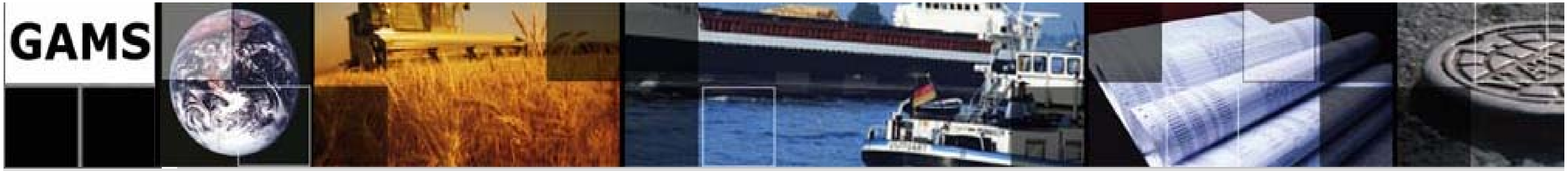
- GAMS/CPLEX Option **`dumptree n`** creates *n* bound files

How difficult is a sub-problem?

- What is a good estimate for how difficult a sub-problem is?
  - Look at the LP value of a sub-problem
    - The smaller the LP value (assuming minimization) the more difficult the sub-problem



- **Cplex Default**
- **Cplex Strong Branching**
- **Spend more time in sub-problem generation**



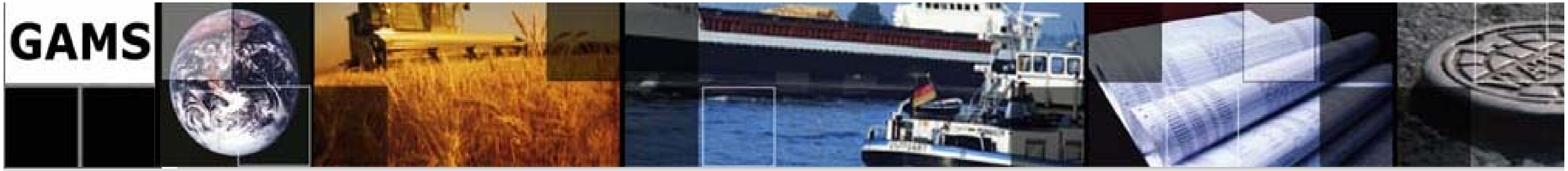
Putting it all together

```
Generate n sub-problems using GAMS/CPLEX with dumpopt n;

loop(n,
  load nth bound file;
  generate and submit nth sub-problem
);

Repeat
  loop(n$(not collected),
    if (n finished,
      load nth-solution and mark n as collected));
  sleep some time;
Until all collected;
```

Communication & Strategy

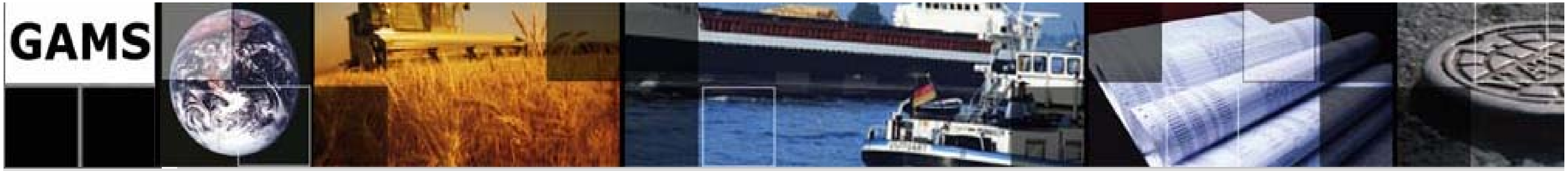
- An incumbent solution allows to prune nodes with larger LP solution value in all sub-problems.
- Hence communicate a newly found incumbent to all sub-problems
  - Sub-problems not started: Start with a **`cutoff`**
  - Running sub-problems: Update the **`cutoff`** with a GAMS/CPLEX option file that is read while running

- Strategy:
  - Have one machine working on good solutions (e.g. CPLEX **`mipemphasis 1`** or **`4`**) using original problem
  - Sub-problems emphasize on best-bound (e.g. CPLEX **`mipemphasis 3`**)



Some results

	ROLL3000	A1C1S1	TIMTAB2* * Added problem cuts
#sub-problems	986	1089	3320
objective	12890	11768.2	1.10656e+06
#Cplex B&B nodes	400,034	1,921,736	17,092,215
CPU time used	50h	3432h	2384h
CPU time wasted	0.5h	248h	360h
Wall time	Over night	Over night	Over night



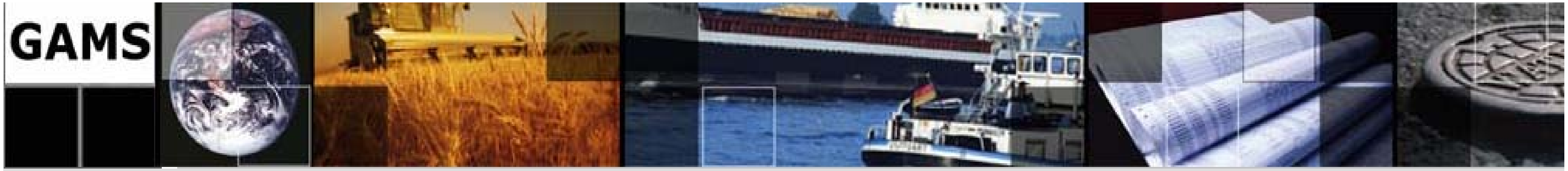
Other Results

- Problem SWATH (TSP type problem)

Sub-problems:	2598 (578 still outstanding)
Objective:	467.407
CPU time used:	6590h
CPU time wasted:	4995h
Nodes explored:	38,012,523

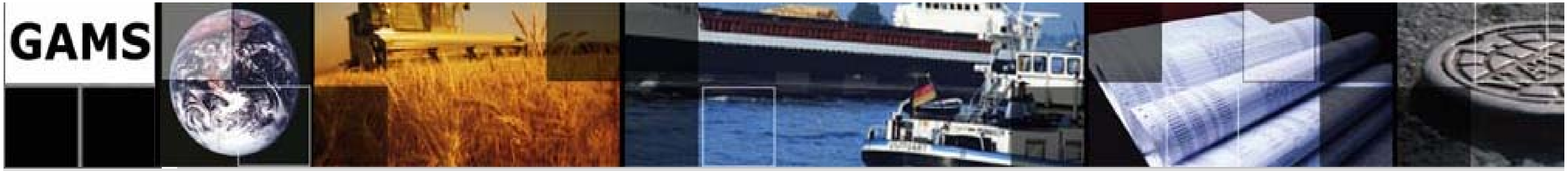
- Second Level Partitioning (subdivide **one** of the 578 outstanding problems [a *difficult* one]):

Sub-problems:	702 (264 still outstanding)
CPU time used:	30600h (3.5 years!)
CPU time wasted:	46344h (5 years!)
Nodes explored:	752,713,119



A word of caution

- Go back to original SWATH paper!
- Understand underlying (20 city) TSP with "supernodes"
- 5 rounds of subtour elimination cuts, 32 extra constraints in all
- Problem solved in less than 20 minutes on a single machine using CoinCbc!



Summary

- GAMS/CPLEX **`dumpopt`** ***n*** to find a-priori problem partition of a MIP
- Using GAMS Grid Facilities, Condor, and GAMS/CPLEX to generate, submit, and solve *n* sub-problems
- Communication of updated incumbent is essential
- Solved two previously unsolved problems (ROLL3000, A1C1S1) from MIPLIB2003 over night (with few hundred machines available)
- Brute force has it’s limits, but with some additional problem specific knowledge (turned into problem specific cuts) one more problem (TIMTAB2) could be solved over night.
- Work on the model level rather than the matrix level
- Some problem in MIPLIB3 will remain unsolved (for a while)