GAMS, Condor and the Grid

Michael C. Ferris
University of Wisconsin

Michael Bussieck, GAMS Corp.

Use of Grid Computation in Optimization

- Aid search for global solutions (typically in non-convex or discrete setting)
  - Pattern search, evolutionary algorithms
  - Branch and bound/cut
- Treat uncertainty (sampling)
- Enhance speed of computation
  - Decomposition approaches
    - Splitting, Benders, Dantzig-Wolfe, Lagrangian
  - Linear algebra

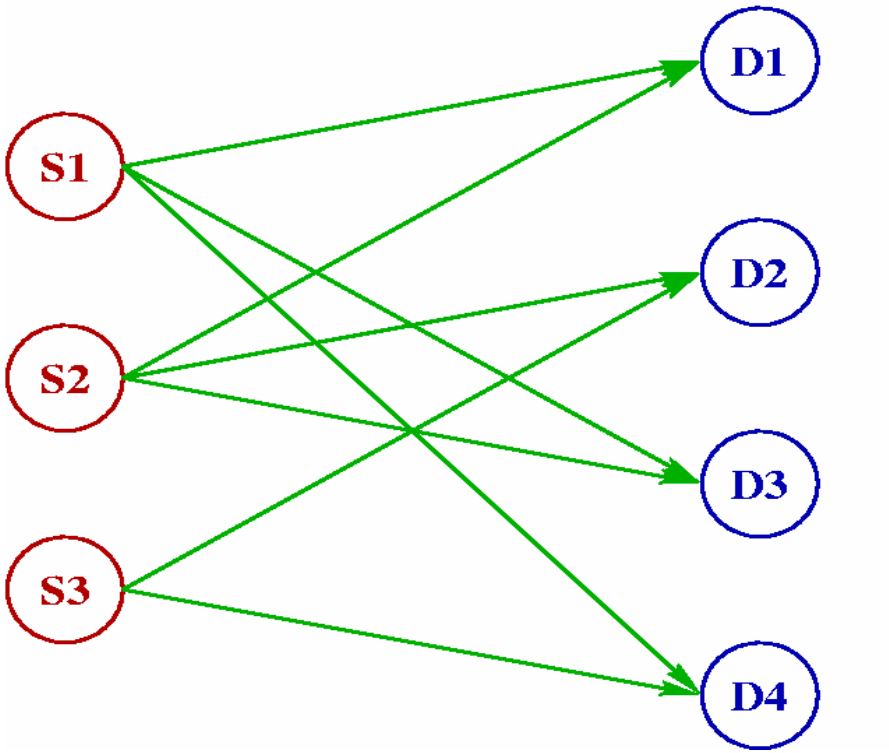
Assumptions for talk

- Hard Optimization Problem to solve
- Access to modeling system
  - GAMS (iccoptlic.txt)
  - AMPL
- Access to Grid computing
  - Condor
  - Sun N6 Grid Engine, Globus
- How to use Grid effectively to solve problem already modeled
- Build on shoulders of giants…

Can we use it effectively?

- High throughput not high performance computing (modify perspective)
- New modeling features of GAMS facilitate use of grid computation and sophisticated solvers
- Optimization expertise shared with computational engines

Transportation model



$$\min_x \quad z = \sum_{(i,j)} c_{i,j} x_{i,j}$$

$$\text{st} \quad \sum_j x_{i,j} \leq a_i, \ \forall i$$

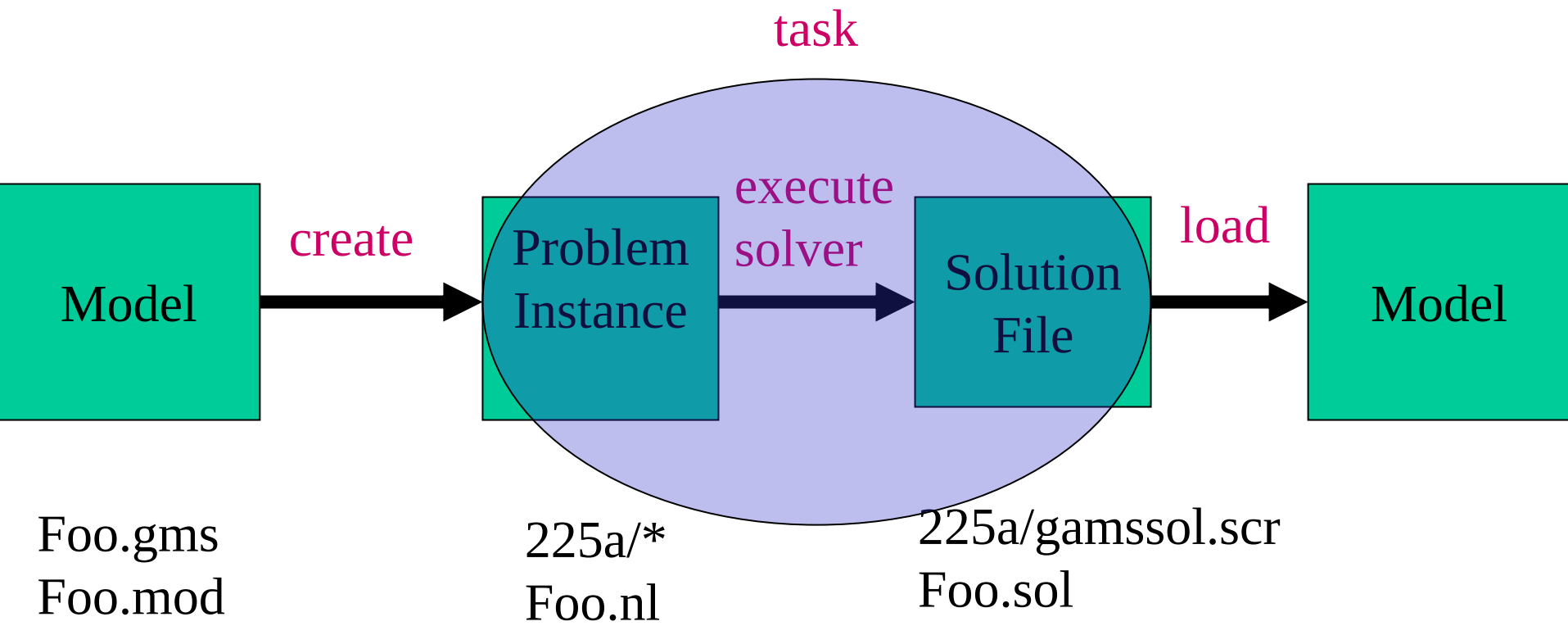
$$\sum_i x_{i,j} \geq b_j, \ \forall j$$

Typical Application for GAMS

```
b(j) = dem(j);
solve transport min z using lp;
report = z.l;
```

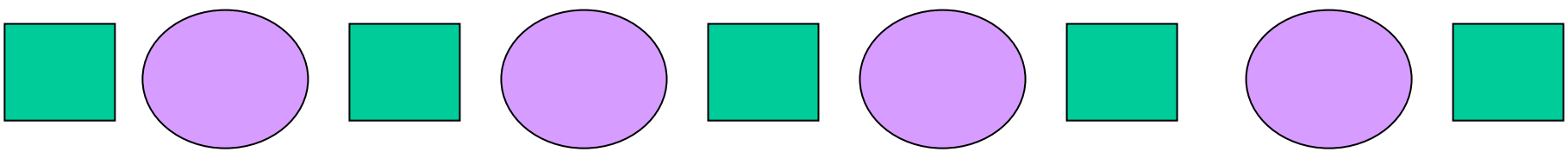
Understand "solve" statement

```
b(j) = dem(j);
solve transport min z using lp;
report = z.l;
```



Typical Application for GAMS

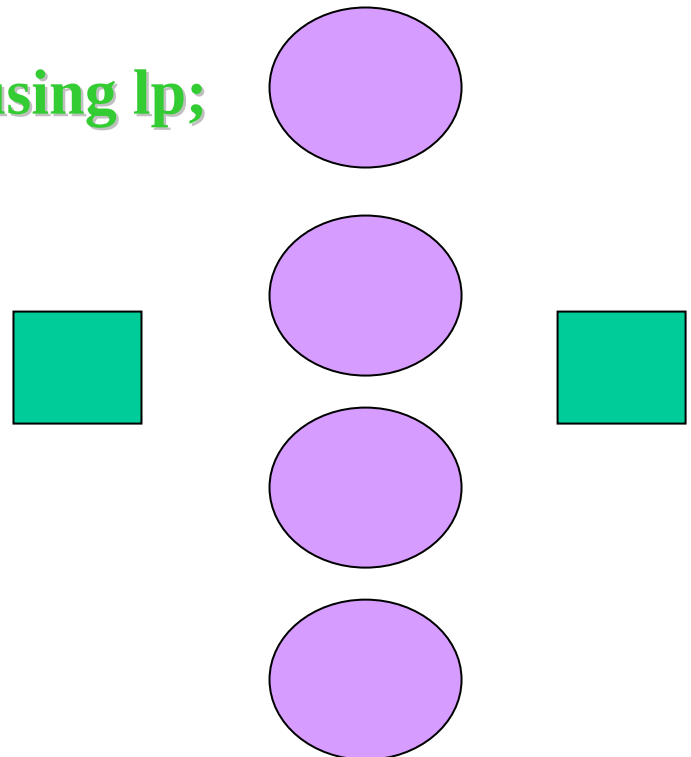
```
loop(s,
   b(j) = dem(s,j)
   solve transport min z using lp;
   report(s) = z.l;
) ;
```



Typical Application for GAMS

Need notion of a handle

```
loop(s,
   b(j) = dem(s,j)
   solve transport min z using lp;
   report(s) = z.l;
) ;
```



Typical Application for GAMS & Grid

```
transport.solvelink = 3;          // turn on grid option
loop(s,
   b(j) = dem(s,j)
   solve transport min z using lp;
   h(s) = transport.handle );    // save instance handle

repeat
   loop(s$handlecollect(h(s)),
      report(s) = z.l;
      h(s) = 0 ) ;    // indicate that we have loaded the solution
   display$sleep(card(h)*0.2) 'was sleeping for some time';
until card(h) = 0 or timeelapsed > 10;
```

Demonstration (source setit)

- cd gams
- gams trnsgrid
- setenv USECONDOR lnx
- gamskeep trnsgrid
- condor_q
- mkdir gdir
- gams trnsgrid gdir=gdir

Demonstration (2)

- gams trnsspawn gdir=gdir s=T
- condor_q
- condor_q
- …
- gams trnscollect gdir=gdir r=T

Demonstration (3)

- setenv USECONDOR mw
- setenv MWWORKERS 2
- gams trnsspawn gdir=gdir s=T
- condor_q
- condor_q
- …
- gams trnscollect gdir=gdir r=T

Exercises

- Change number of scenarios
- Change solver from default
- Run the danwolfe example using mw and the same gdir
- Read last few lines of 'mcpgrid.gms' and 'mcpcollect.gms' and run this
- gamslib openpit; use mw grid option and update model to run with 8 pits

Solution hints

- option lp=xpress;
- gams danwolfe gdir=gdir
- gams mcpgrid gdir=gdir s=T
- gams mcpcollect gdir=gdir r=T
- gams openpit --pmx=8 solvelink=3 gdir=gdir

Multiple Solvers/Platforms

- Can use all supported solvers including:
  - CPLEX, XPRESS, PATH, SNOPT, MOSEK
- Runs on multiple platforms using heterogeneous machines for solvers
- Can interleave solutions on host and worker
- Available right now!

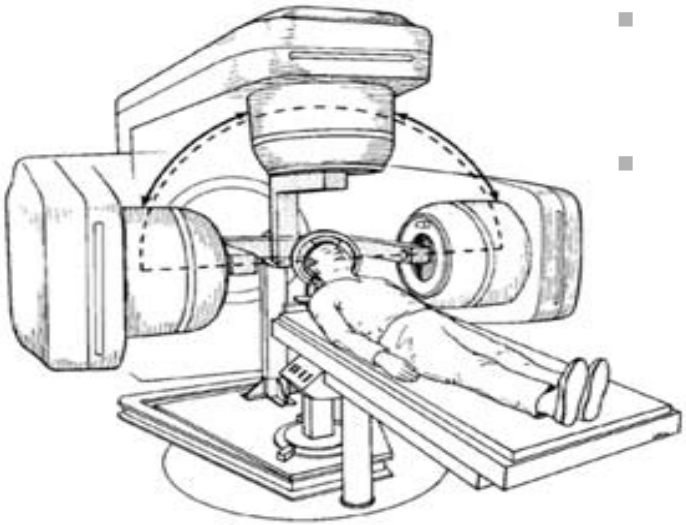
Clean up mw!

- Either condor_rm your mw "server" job
- Or "mv gdir foo; sleep 20; mv foo gdir

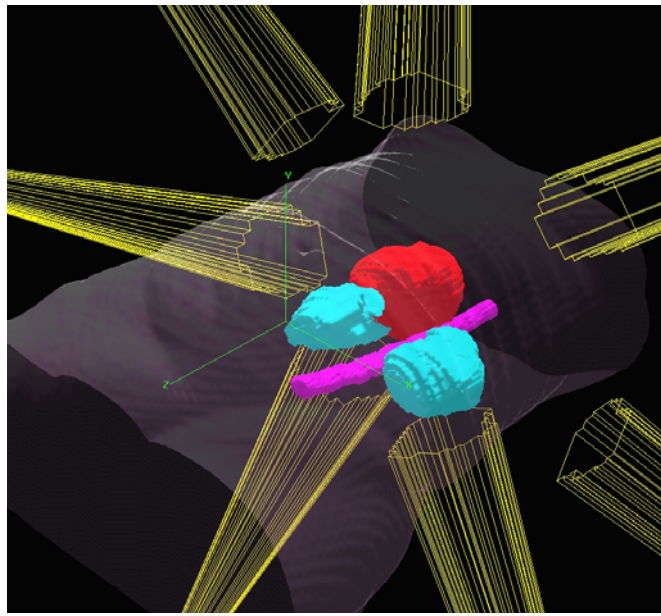
Feature Selection

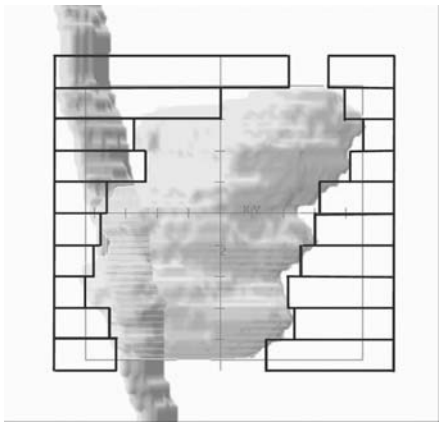
- Select best features for classification
- Evaluate with 10-fold cross validation
- Perform validation multiple times
  - Reduce variance
  - Obtain better estimate
- Each validation creates 10 jobs
- Perform 20 concurrent validations
  - Generates 200 independent problems
  - Each problem is an integer program

Radiotherapy Treatment

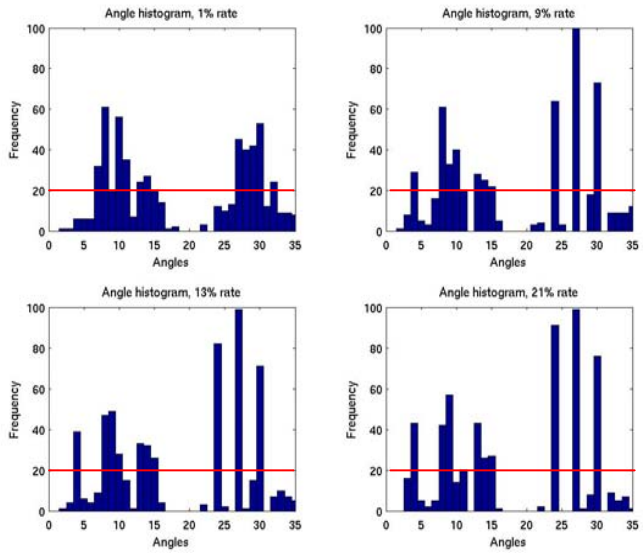


- Fire from multiple angles
- Superposition allows high dose in target, low elsewhere



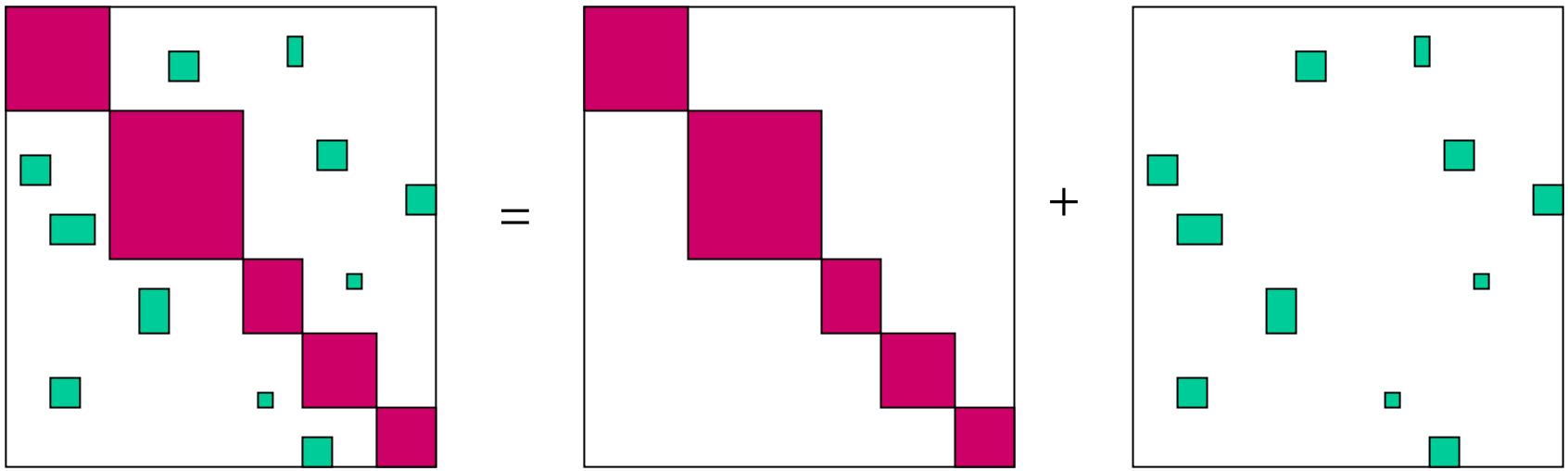


- Beam shaping via collimator
- Other enhancements
- Sampling allows good angles to be determined quickly and in parallel



Trade/Policy Model (MCP)

- Split model (18,000 vars) via region



- Gauss-Seidel, Jacobi, Asynchronous
- 87 regional subprobs, 592 solves

Gauss-Seidel

```
loop(iter$(not done),
  loop(p,
    r(i) = yes$inp(i,p);
    x.fx(i)$(not r(i)) = x.l(i);
    solve trademod using mcp;
    x.lo(i) = 0; x.up(i) = xup(i) );
);
```

Jacobi

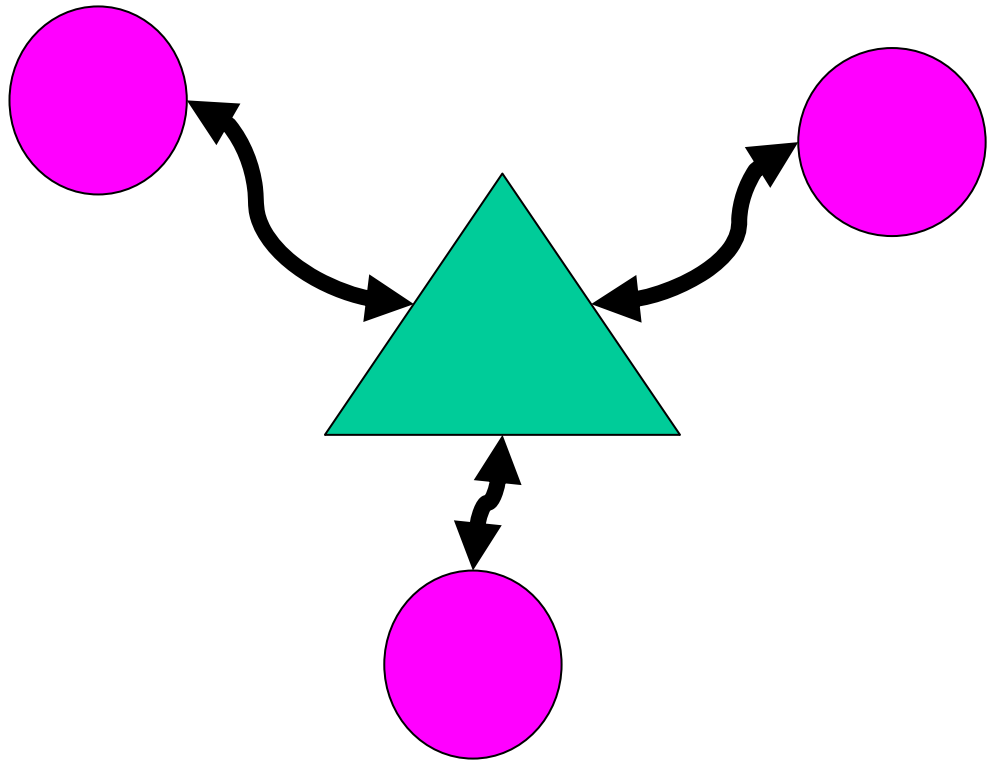
```
gemtap.solvelink = 3;
loop(iter$(not done),
  loop(p,
    r(i) = yes$inp(i,p);
    x.fx(i)$(not r(i)) = x.l(i);
    solve gemtap using mcp;
    x.lo(i) = 0; x.up(i) = xup(i);
    h(p) = gemtap.handle );

  repeat; loop(p$h(p),
    if(handlecollect(h(p)),
       h(p) = 0; ););
  until card(h) = 0;
);
```

Asynchronous

```
gemtap.solvelink = 3;
repeat; loop(p$h(p),
  if ( handlecollect(h(p)),
    h(p) = 0;
    if (sum(k, dev(k)) > tol,
        loop(k$(h(k) eq 0 and dev(k),
            gemtap.number = ord(k)-1;
            r(i) = yes$inp(i,k);
            x.fx(i)$(not r(i)) = x.l(i);
            solve gemtap using mcp;
            x.lo(i) = 0; x.up(i) = xup(i);
            h(k) = gemtap.handle;
);););););
until (card(h) = 0);
```

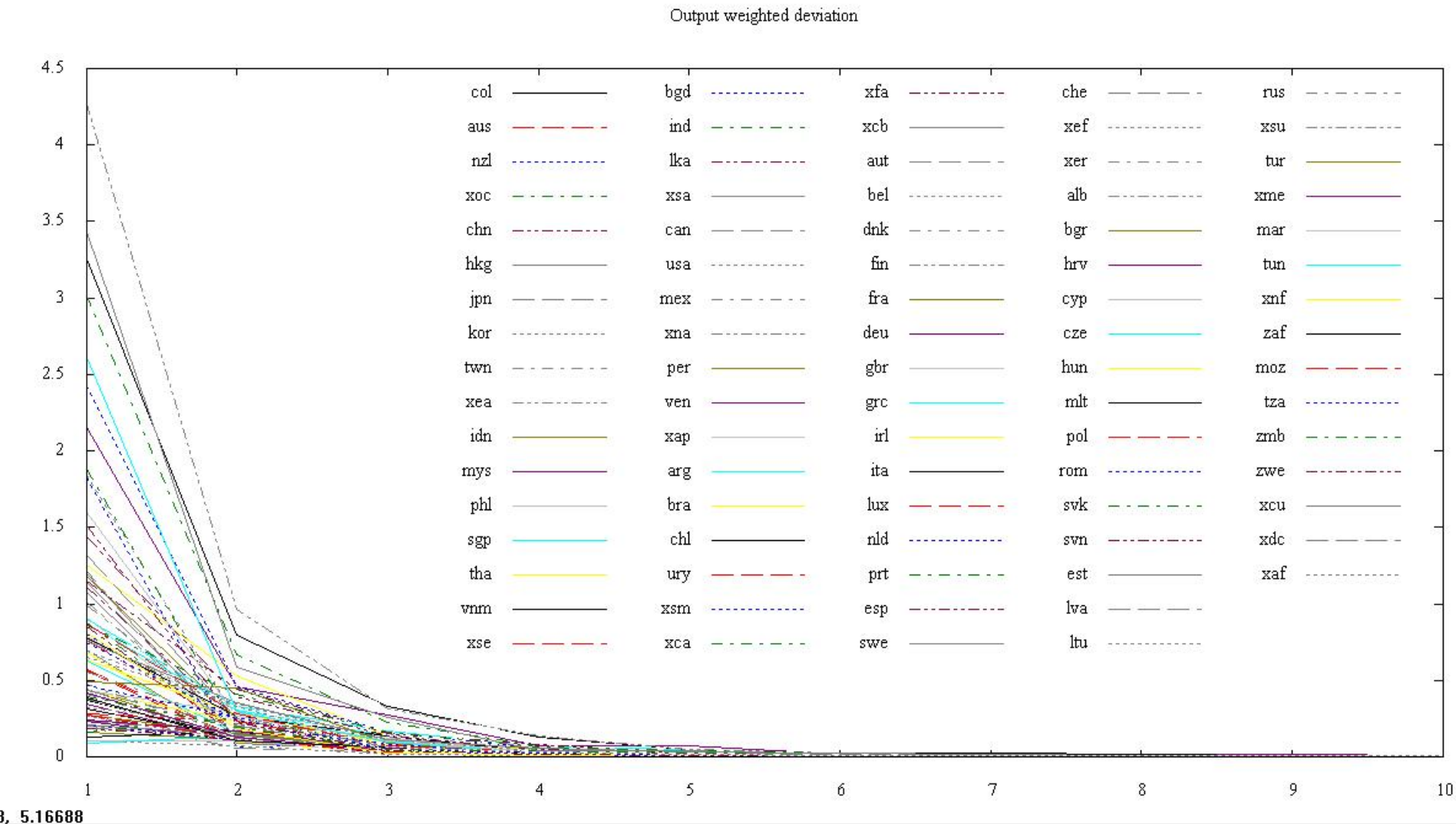
Model knowledge decomposition

- Pink model - open economy (regions)
- Green model - (partial) spatial equilibrium (commodities)
- Links are imports and exports



Calibrate supply and demand functions to points, and communicate functional forms, not points

Deviations by iteration



3, 5.16688

GAMS/Grid

- Commercial modeling system – abundance of real life models to solve
- Any model types and solvers allowed
  - Scheduling problems
  - Radiotherapy treatment planning
  - World trade (economic) models
  - Sensitivity analysis
  - Cross validation, feature selection
- Little programming required
- Separation of model and solution maintained

The interface

- handlecollect(h)
  - loads in solution data from model indexed by handle
- handledelete(h)
  - deletes grid directory
- handlesubmit(h)
  - reruns task without regenerating input data
- h = modelname.handle
- modelname.number
  - instance number used to generate next handle
- handlestatus(h)
- execute_loadhandle modelname;
- gdir

Under the hood…

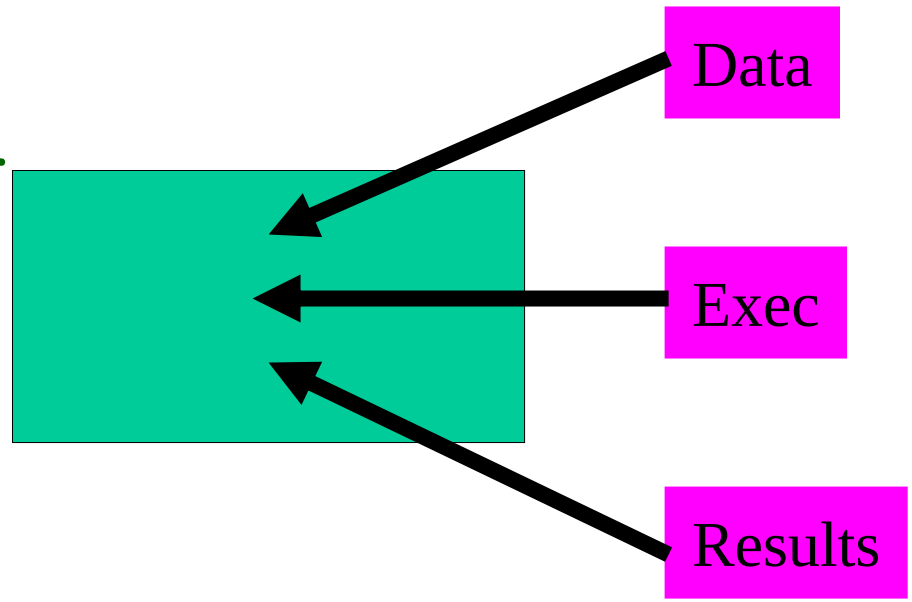
- Using "solvelink=3" each solve statement generates a new "gridxxx" subdirectory
- An executable/script "gmsgrid" is run for each solve
  - Runs "solver" as background process
  - Modified to submit condor job or set up and run "MW server"
  - Creates "re-run" script in case of failure

MW-GAMS

- Generate "gams task" worker
- Data common to all tasks is sent to workers only once
- (Try to) retain workers until the whole computation is complete—don't release them after a single task
- Master and worker executables already made - modeler just flips a switch!
  - USECONDOR=mw

Worker / task

- Local copy of gams needed
  - Zip file, job dir
  - Mimic environment
- Problem instance
- Start flag
- End flag
- Trigger file
  - Updates



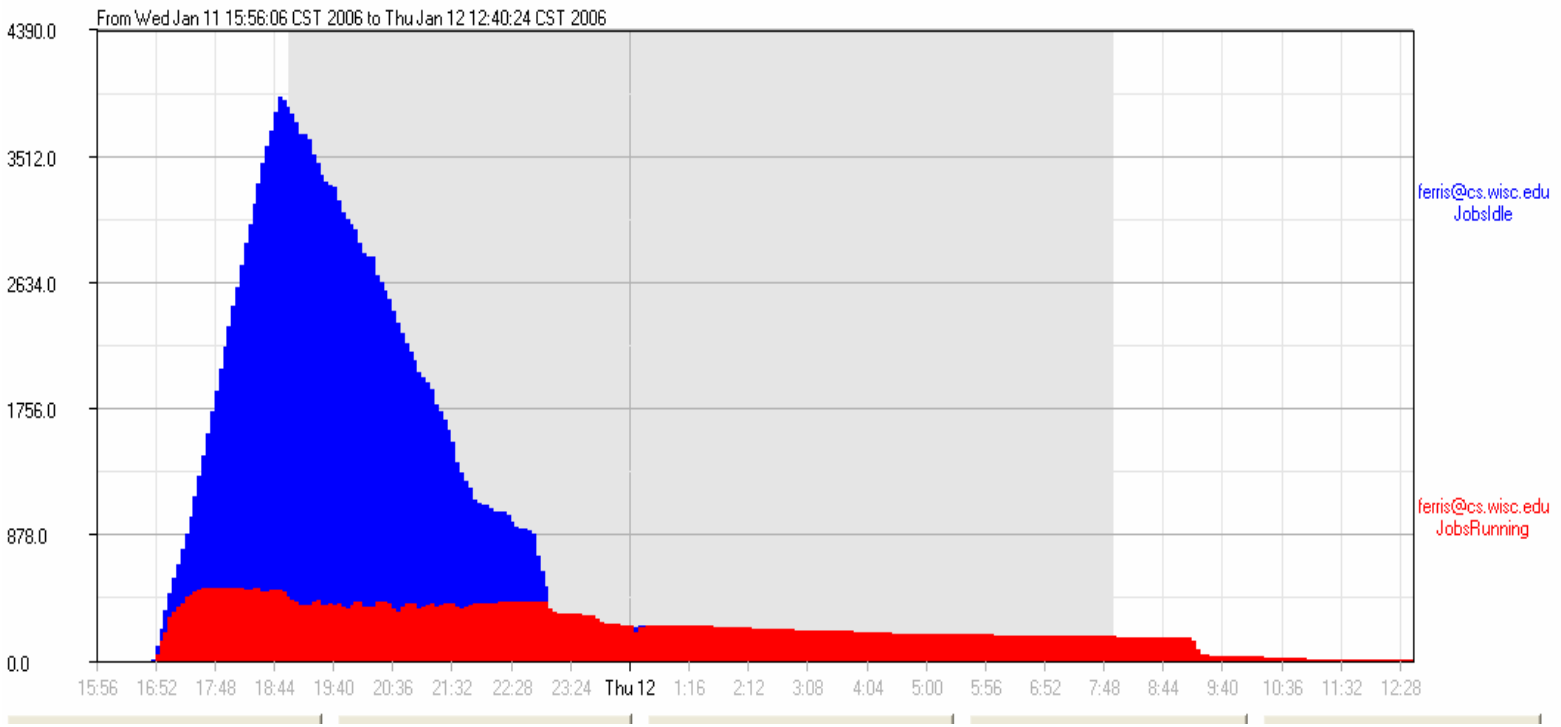
Shortcomings/Future Work

- Iterative scheme updates small amount of model "data"
- As convergence occurs (prove it!) models become easy to solve (great start point)
- Model regeneration time is longer than solution time!
- Fix: use MW and gams_submit

Massively Parallel MIP

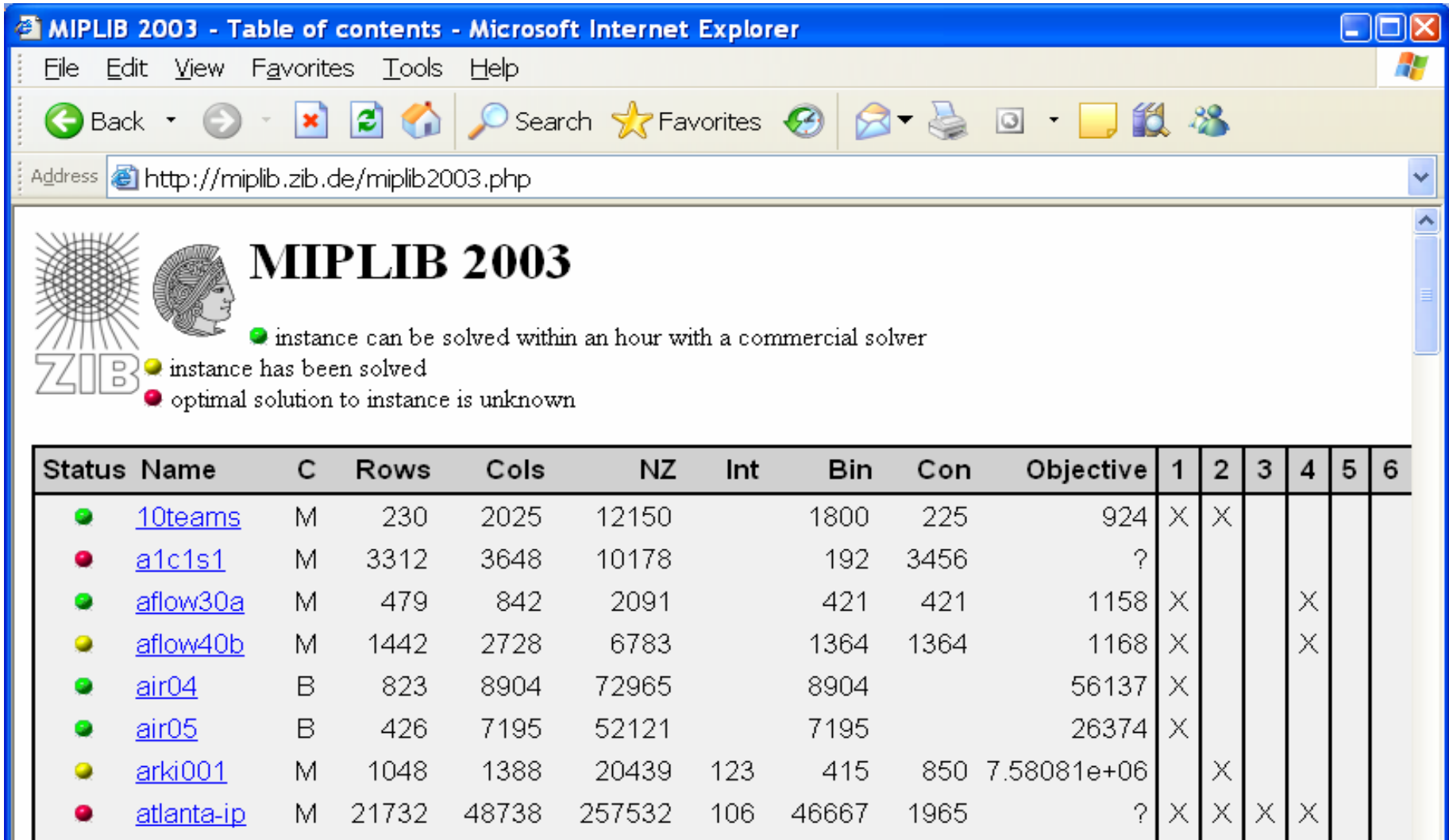
- **MIP/B&C Algorithm ideal to parallelize**
  - Master/Worker Paradigm (process nodes in parallel)
    - Software: FATCOP/Condor, BCP/PVM, PICO/MPI
  - A-priori subdivision into *n* independent problems
    - Seymour problem solved that way
  - Open Pit Mining (openpit in GAMS Model library)
    - Partition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

4096 MIPS on Condor Grid



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

MIPLIB 2003 had 13 unsolved instances



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 - CPLEX
- 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 machine resources

Problem with a-priori partitioning

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

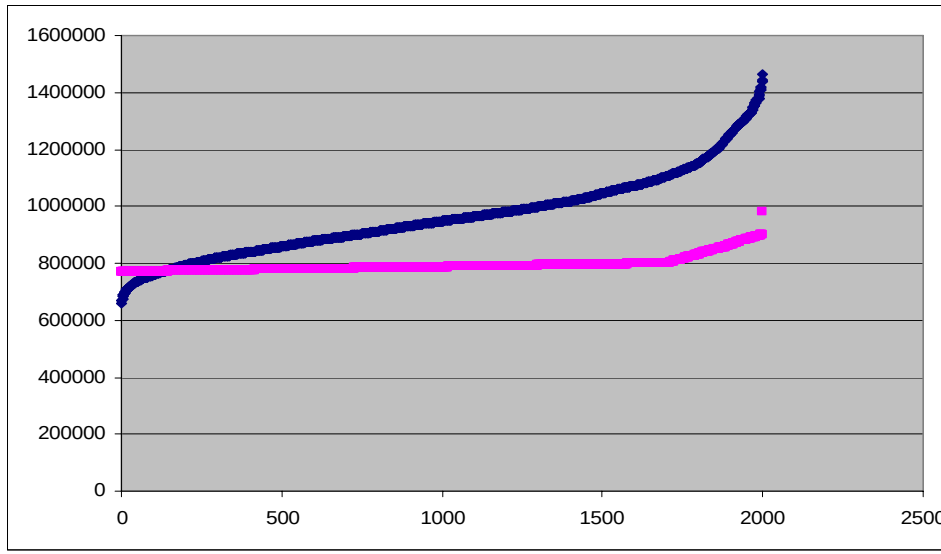
- How can we find *n* sub-problems with similar (but reduced) level of difficulty?
  - B&C Code keeps a list of *open/unexplored* nodes
  - Problem-bounds of these open nodes represent partitioning of the original problem

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 subproblem?

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



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

Putting it all together

```
Generate n sub-problems using GAMS/CPLEX with dumptree 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

- Incumbent solution allows pruning of nodes with larger LP solution value
  - Use BCH facility to dump out new incumbents in each worker whenever found in a subproblem
- Communicate newly found incumbent to all workers
  - Subproblems not started: Start with **`cutoff`**
  - Running subproblems: Update **`cutoff`** with a GAMS/CPLEX option file that is read while running (solver option facilitates on-the-fly strategy changes)

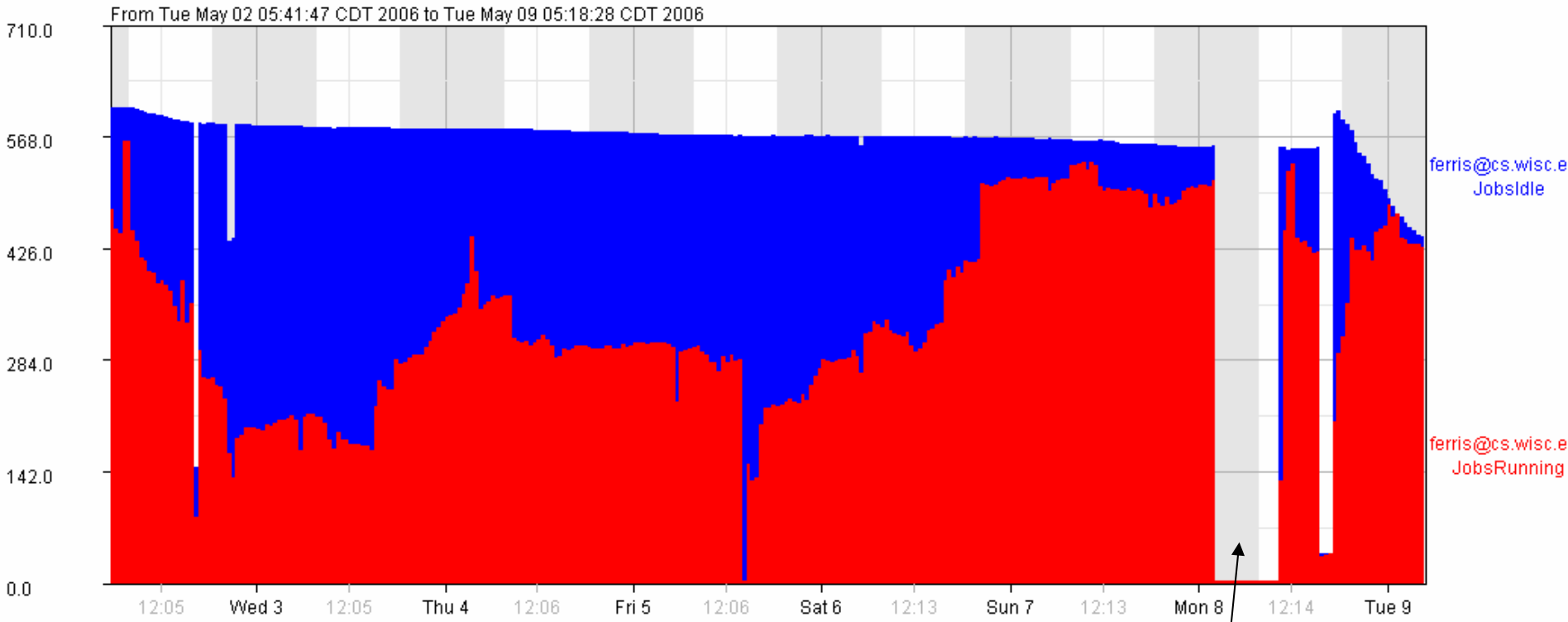
Worker communication

- If have shared file system, just use trigger files
- condor_chirp is a utility that helps mimic the shared file system
  - Provides fetch, remove, put
- Workers use this to communicate with master/submit machine

Strategy

- Strategy:
  - Have one machine working on good solutions for original problem
    - CPLEX `mipemphasis 1` or `4`
  - Subproblem emphasis on best-bound
    - CPLEX `mipemphasis 3`
  - Repartition longest running jobs
  - Restart from incumbent (cf NLP)

Grid resources used



Partitioned into 1000 subproblems, over 300 machines running for multiple days

main submitting machine died, jobs not lost

Some results

	ROLL3000	A1C1S1	TIMTAB2 (added problem cuts)
#sub-problems	986	1089	3320
objective	12890	11503.4	1096557.
#Cplex B&B nodes	400,034	1,921,736	17,092,215
CPU time used	50h	3452h	2384h
CPU time wasted	0.5h	248h	361h
Wall time	Over night	Over night	Over night

Other Results

- Problem SWATH (TSP type problem)
  + sub-tour elimination cuts:

- Subproblems: 1539 (23 not finished)
- Objective: 467.407
- CPU time used: 36159 hr (4.1 years)
- CPU time wasted: 71557 hr (8.2 years!)
- Nodes explored: 721,718,141

- Second Level Partitioning (subdivide of several of the 23 outstanding problems):

Subproblems: 2000
CPU time used: 2,232 hr
CPU time wasted: 24,000 hr
Nodes explored: 464,006,423

A word of caution

- Go back to original SWATH paper!
- Understand underlying (20 var) 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!

Scheduling Multistage Batch Plants

- Solution within 1 day
- Three level decision process (GAMS)
  - Split order into batches
  - Assign batches to processing units
  - Sequence batches over stages
- Instance 1: solved sequentially CPLEX
- Instance 2: solved GAMS/CPLEX/Condor
- Instance 3: gap (1176-1185) after 24h

Adaptive SB Method

- Split model using "domain expertise" at top levels
  - 234 jobs, fixes batches and some assignments
- Apply (very) strong branching to generate a collection of subproblems
- Solve each subproblem
  - If 2 hour time limit reached, reapply strong branching to subdivide and resolve
- Instance 3 solved (22 hours) - 4 branching levels
- (5 days,22 hrs; nodes = 58,630,425; 7356 jobs)

Summary

- GAMS/CPLEX dumptree n
  - a-priori problem partition of MIP
- Use GAMS Grid facilities, Condor, and GAMS/CPLEX to generate, submit, and solve n subproblems
- 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 its limits, but with some additional problem specific knowledge (turned into problem specific cuts) one more problem (TIMTAB2) could be solved over night
- Problem knowledge still very useful, solved (SWATH)
- Some problems in MIPLIB2003 will remain unsolved for a while

Conclusions

- Grid systems available (e.g. Condor, IBM, SUN)
- Grid computing convenient via simple language extensions to modeling languages
- Can experiment with coarse grain parallel approaches for solving difficult problems
- Exploiting underlying structure and model knowledge key for "larger, faster" solution

- Easy, adaptive, improves, need expertise

Future extensions

- "Time-constrained" problem solution (as opposed to "real-time")
- Re-optimization (model updating)
- Global optimization
- Commercial use
- Saving intermediate solution results
- Further application deployment