GAMS, Condor and the Grid: Solving Hard Optimization Models in Parallel

Michael C. Ferris
University of Wisconsin
Michael Bussieck, GAMS Corp.

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 distributively owned
- Licensing, communication and security issues

Condor Features

- Uses dedicated clusters and cycles from desktop workstations (> 1000 machines available for "ferris")
- Heterogeneous machines, with or without shared file system
- Machines updated regularly
- Fault tolerance
  - Jobs submitted are eventually executed
- Available for download, configurable

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

Typical Application for GAMS

```
demand = 42; cost = 14;
solve mymodel min obj using minlp;
report = var.l;
```

Typical Application for GAMS

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

Typical Application for GAMS & Grid

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

repeat
  loop(scenario$h(scenario),
    if(handlestatus(h(scenario)),
      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;
```

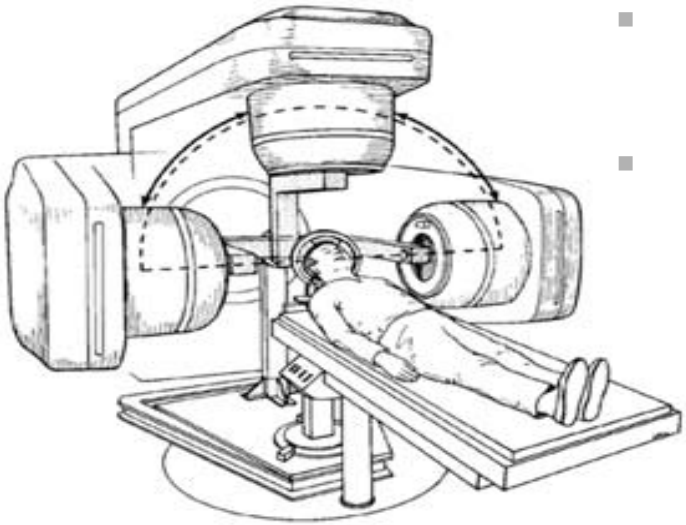
Multiple Solvers/Platforms

- 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, maintains data confidentiality
- Available right now!

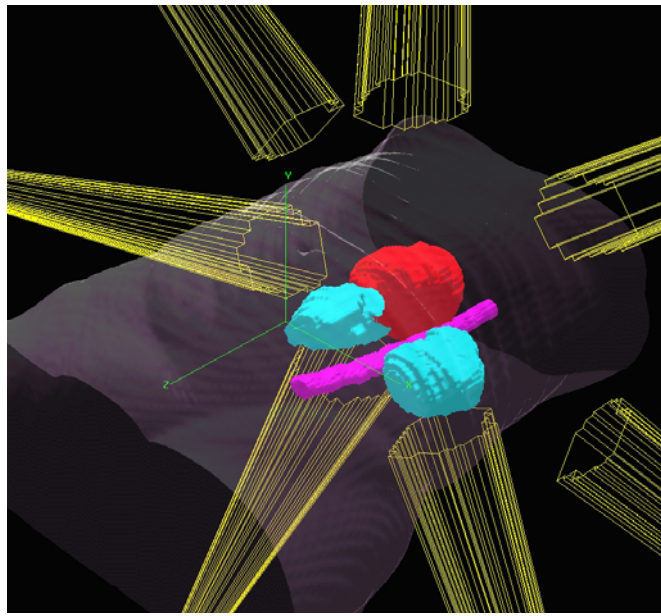
Feature Selection

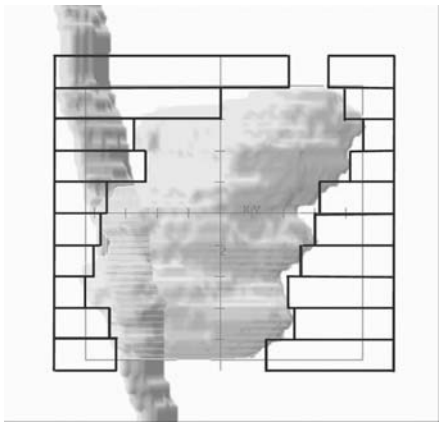
- Select best features for classification
- Evaluate with 10-fold cross validation
- Perform validation multiple times
  - Reduce variance
  - Obtain better estimate
- Each validation creates 10 jobs
- Perform 20 concurrent validations
  - Generates 200 independent problems
  - Each problem is an integer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

Radiotherapy Treatment

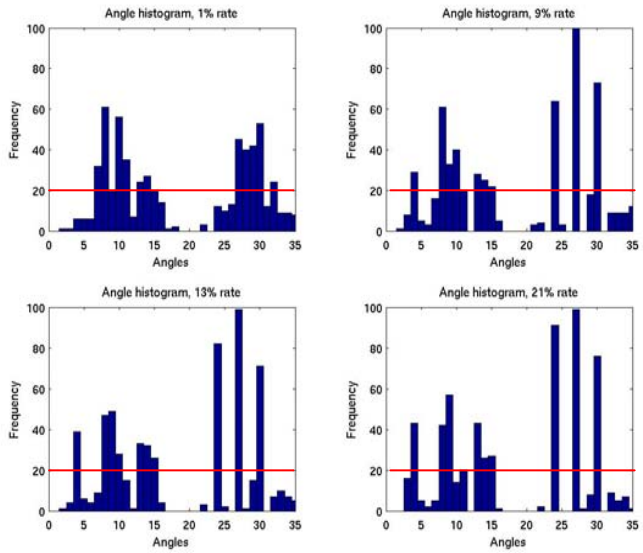


- Fire from multiple angles
- Superposition allows high dose in target, low elsewhere





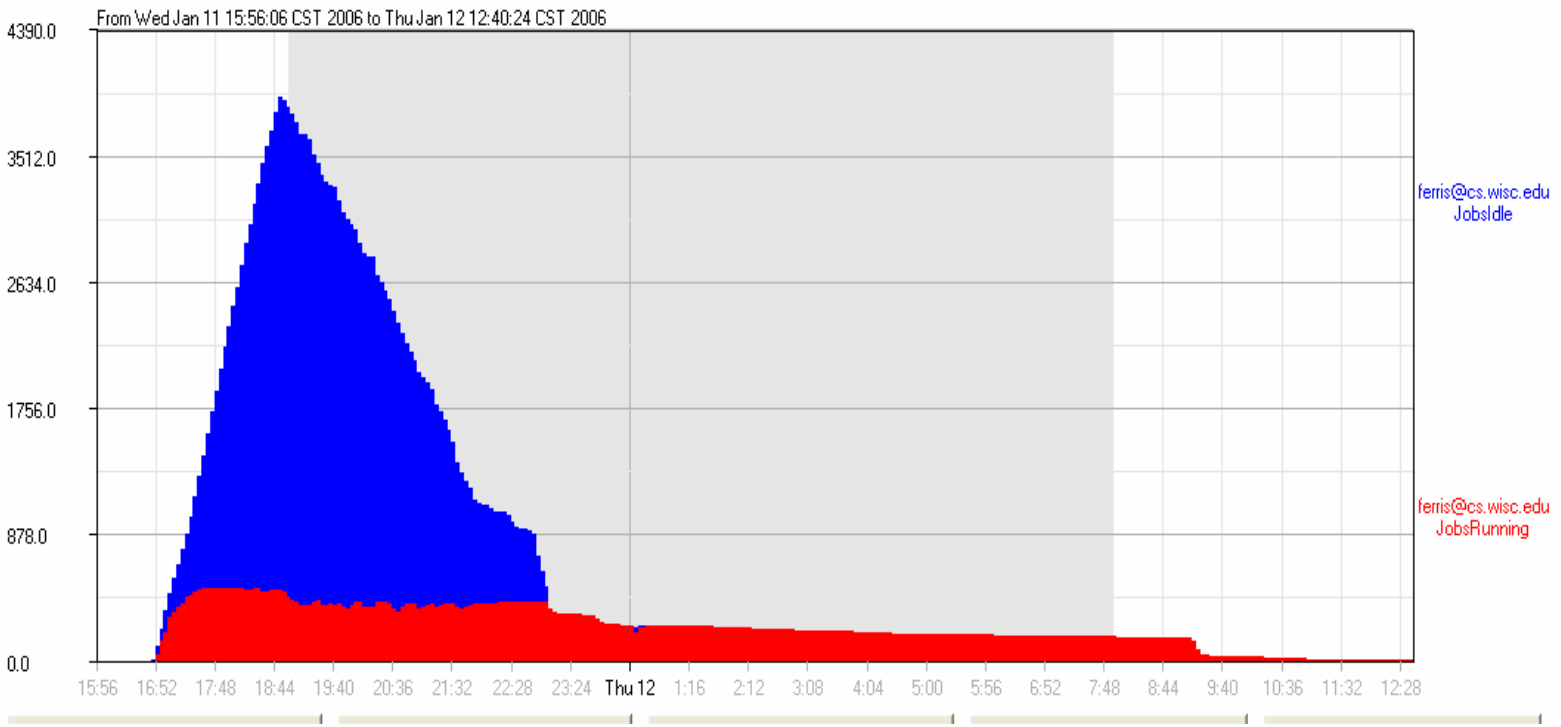
- Beam shaping via collimator
- Other enhancements
- Sampling allows good angles to be determined quickly and in parallel



Massively Parallel MIP

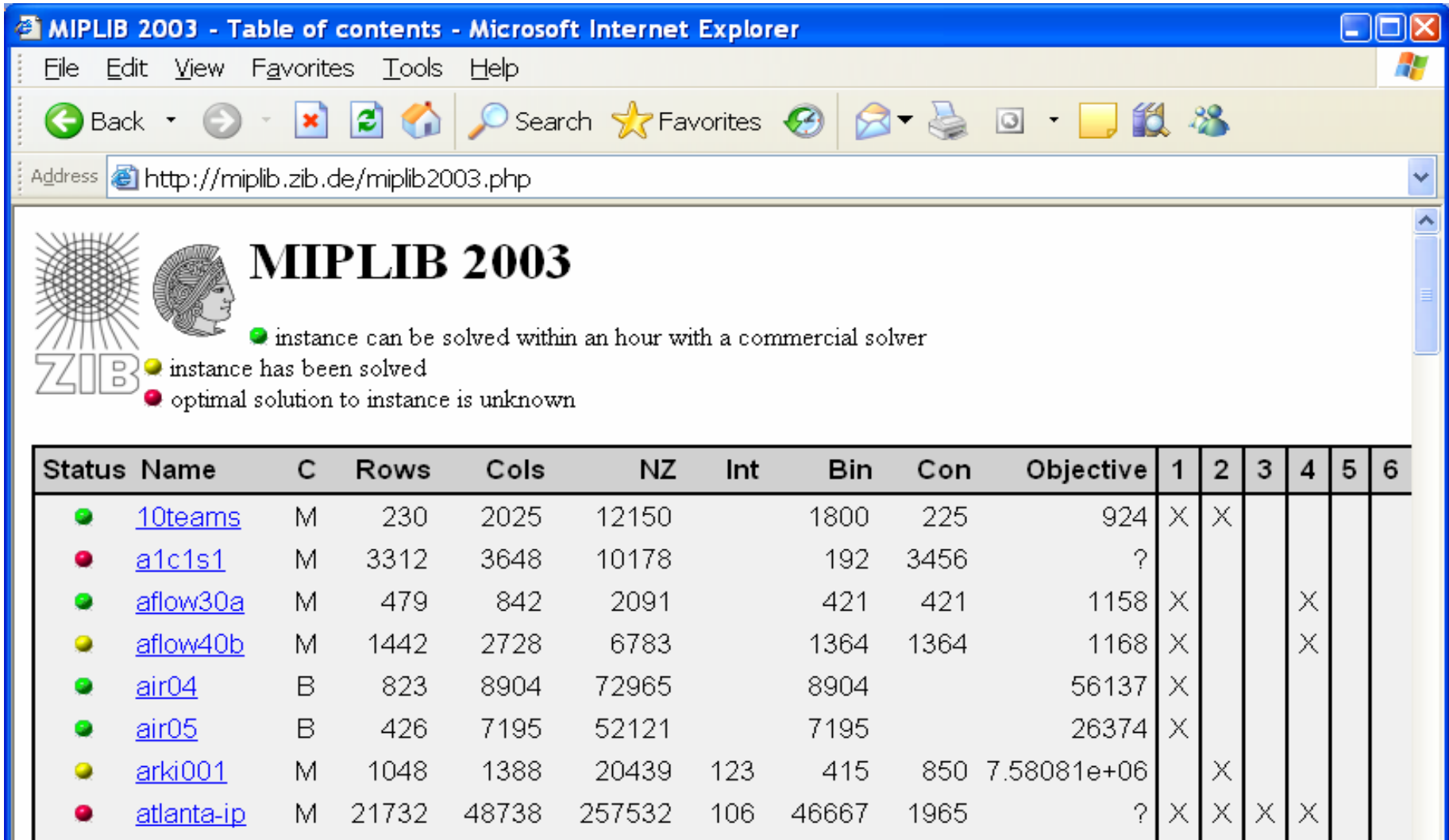
- MIP/B&C Algorithm ideal to parallelize
  - Master/Worker Paradigm (process nodes in parallel)
    - Software: FATCOP/Condor, BCP/PVM, PICO/MPI
  - A-priori subdivision into *n* independent problems
    - Seymour problem solved that way
  - Open Pit Mining (openpit in GAMS Model library)
    - Partition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

Problem with a-priori partitioning

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

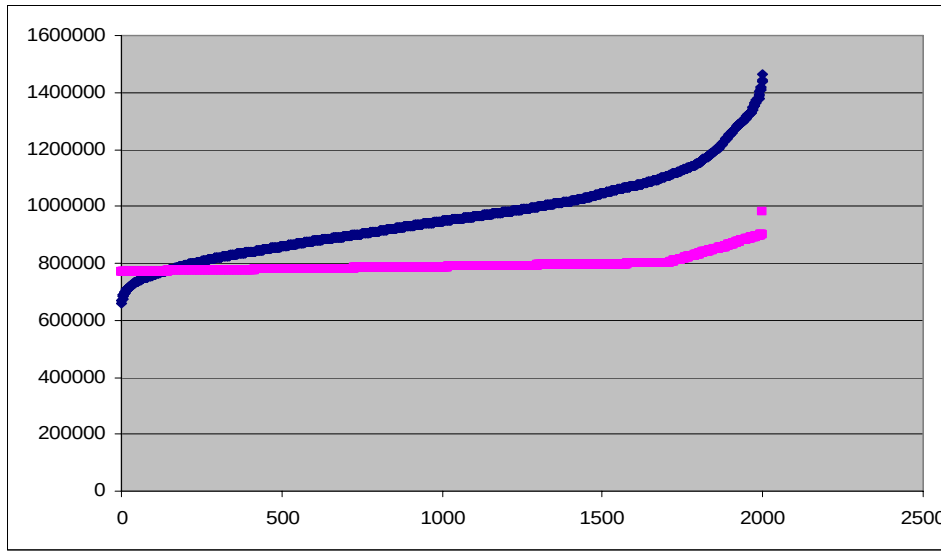
- How can we find *n* sub-problems with similar (but reduced) level of difficulty?
  - B&C Code keeps a list of *open/unexplored* nodes
  - Problem-bounds of these open nodes represent partitioning of the original problem

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 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;
```

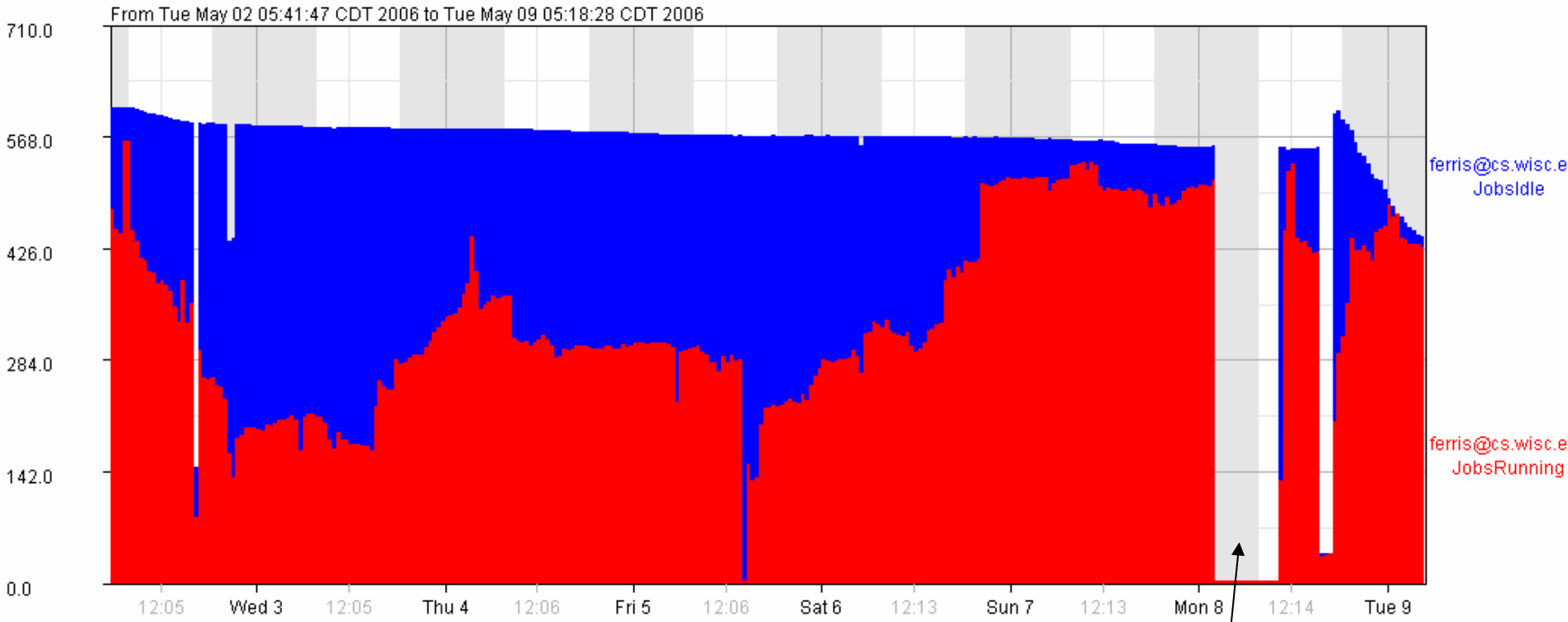
Communication

- Incumbent solution allows pruning of nodes with larger LP solution value
  - How greedy are you?
  - Shared file system (unreliable, unavailable)
  - condor_chirp for inter-worker communication (background process on worker)
- Hence communicate newly found incumbent to all subproblems
  - 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)

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 dumpopt 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

- Massive parallel and distributed computing environments are available (e.g. Condor, IBM, SUN)
- Grid computing capability available for optimizers in convenient environment via simple language extensions to modeling languages
- Today's modeling languages are well suited to experiment with coarse grain parallel approaches for solving difficult problems

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