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 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 Algebraic Modeling System:

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
mymodel.solvelink=3;
loop(scenario,
    demand=demand(scenario); cost=cost(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 4096 sub-problems
  - Experiments (Ferris) at UW using Condor Pool
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!
Information on past Condor Week meetings

Current Releases
Stable series: Condor Version 8.6.11 released March 28th, 2006
Development series: Condor Version 6.7.20 released June 22nd, 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

<table>
<thead>
<tr>
<th>Status</th>
<th>Name</th>
<th>C</th>
<th>Rows</th>
<th>Cols</th>
<th>NZ</th>
<th>Int</th>
<th>Bin</th>
<th>Con</th>
<th>Objective</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>•</td>
<td>10teams</td>
<td>M</td>
<td>230</td>
<td>2025</td>
<td>12150</td>
<td>1800</td>
<td>225</td>
<td></td>
<td>924</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>•</td>
<td>a1c1s1</td>
<td>M</td>
<td>3312</td>
<td>3648</td>
<td>10178</td>
<td>192</td>
<td>3456</td>
<td>?</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>•</td>
<td>aflow30a</td>
<td>M</td>
<td>479</td>
<td>842</td>
<td>2091</td>
<td>421</td>
<td>421</td>
<td>1158</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>•</td>
<td>aflow40b</td>
<td>M</td>
<td>1442</td>
<td>2728</td>
<td>6783</td>
<td>1364</td>
<td>1364</td>
<td>1168</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>•</td>
<td>air04</td>
<td>B</td>
<td>823</td>
<td>8904</td>
<td>72965</td>
<td>8904</td>
<td></td>
<td></td>
<td>56137</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>•</td>
<td>air05</td>
<td>B</td>
<td>426</td>
<td>7195</td>
<td>52121</td>
<td>7195</td>
<td></td>
<td></td>
<td>26374</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>•</td>
<td>arch001</td>
<td>M</td>
<td>1048</td>
<td>1388</td>
<td>20439</td>
<td>123</td>
<td>415</td>
<td>850</td>
<td>7.58081e+06</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>•</td>
<td>atlanta-ip</td>
<td>M</td>
<td>21732</td>
<td>48738</td>
<td>257532</td>
<td>106</td>
<td>46667</td>
<td>1965</td>
<td>?</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Node</th>
<th>Left</th>
<th>Objective</th>
<th>NInf</th>
<th>Integer</th>
<th>Best Node</th>
<th>ItCnt</th>
<th>Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>29.6862</td>
<td>64</td>
<td>29.6862</td>
<td>165</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>37</td>
<td>17.0000</td>
<td>14</td>
<td>25.0000</td>
<td>2230</td>
<td></td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>70</td>
<td>21.8429</td>
<td>22</td>
<td>24.0000</td>
<td>4022</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- GAMS/CPLEX Option `dump tree 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 $n$th bound file;
  generate and submit $n$th sub-problem
);

Repeat
  loop($n$(not collected),
    if ($n$ finished,
        load $n$th-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

<table>
<thead>
<tr>
<th></th>
<th>ROLL3000</th>
<th>A1C1S1</th>
<th>TIMTAB2*</th>
</tr>
</thead>
<tbody>
<tr>
<td>#sub-problems</td>
<td>986</td>
<td>1089</td>
<td>3320</td>
</tr>
<tr>
<td>objective</td>
<td>12890</td>
<td>11768.2</td>
<td>1.10656e+06</td>
</tr>
<tr>
<td>#Cplex B&amp;B nodes</td>
<td>400,034</td>
<td>1,921,736</td>
<td>17,092,215</td>
</tr>
<tr>
<td>CPU time used</td>
<td>50h</td>
<td>3432h</td>
<td>2384h</td>
</tr>
<tr>
<td>CPU time wasted</td>
<td>0.5h</td>
<td>248h</td>
<td>360h</td>
</tr>
<tr>
<td>Wall time</td>
<td>Over night</td>
<td>Over night</td>
<td>Over night</td>
</tr>
</tbody>
</table>
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)