Solving Hard Integer Programs with MW

JEFF LINDEROTH

ISE Department
COR@L Lab
Lehigh University
jtl3@lehigh.edu

2007 Condor Jamboree
Madison, WI
May 2, 2007

Thanks! NSF OCI-0330607, CMMI-0522796, DOE DE-FG02-05ER25694

Collaborators

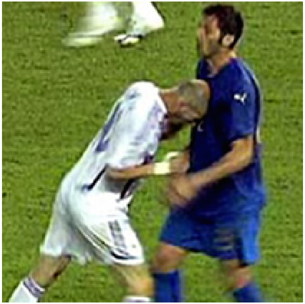
François Margot
Carnegie Mellon

Greg Thain
UW-Madison

In Our Last Episode...

The Design of a Gambling System

- Predict the outcome of v soccer matches
- $\alpha = 3$
  - 0: Team A wins
  - 1: Team B wins
  - 2: Draw
- You win if you miss at most $d = 1$ games



The Football Pool Problem

What is the minimum number of tickets you must buy to guarantee that you hold a winning ticket?

How Many Must I Buy?

Known Optimal Values

v	1	2	3	4	5
$\lvert C_v^* \rvert$	1	3	5	9	27

The Football Pool Problem

What is $\lvert C_6^* \rvert$?

- Despite significant effort on this problem for > 40 years, it is (was) only known that

$$65 \le C_6^* \le 73$$

But It's Trivial!

- There is a simple formulation of the problem as a reasonably-sized integer program (IP)
- For each $j \in W$, let $x_j = 1$ iff the word j is in code C
- Let $A \in \{0,1\}^{|W| \times |W|}$
  - $a_{ij} = 1$ iff word $i \in W$ is distance $\leq d = 1$ from word $j \in W$

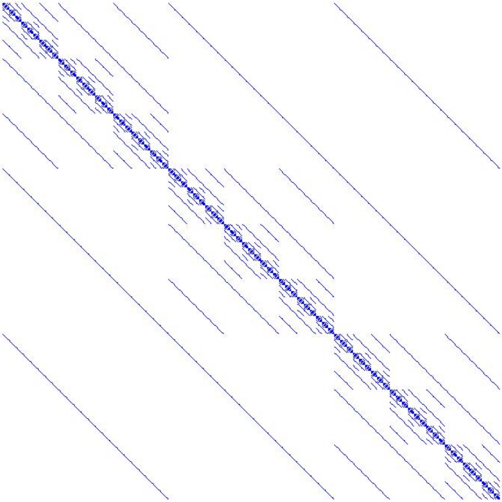
IP Formulation

$$\min e^T x$$

$$\begin{aligned} \text{s.t.} \quad Ax &\geq e \\ x &\in \{0,1\}^{|W|} \end{aligned}$$

COR@L
COMPUTATIONAL OPTIMIZATION RESEARCH AT LEHIGH

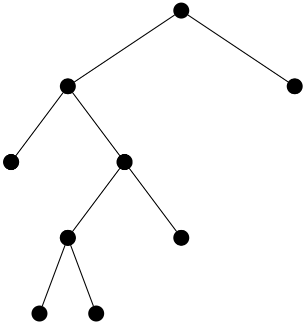
Football Pool Covering Matrix



Beautiful But Deadly

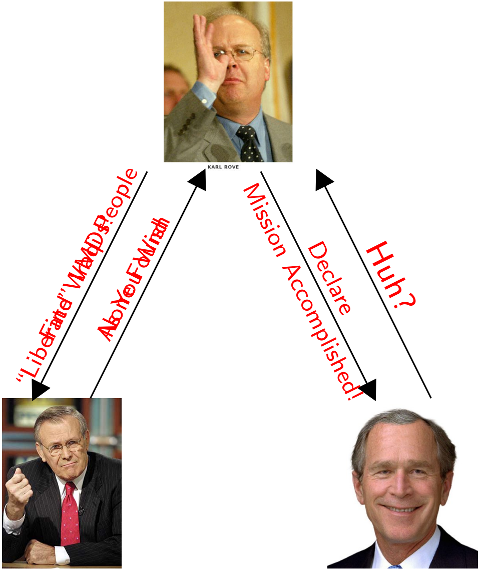
- Workhorse algorithm is a tree-search procedure known as branch-and-bound.
- CPLEX: A commercial IP package that is putting integer programmers out of business.
- CPLEX routinely solves 0-1 integer programs with (tens of) thousands of variables and constraints
- Theorem: "Pretty" Matrices Make Hard IPs
- Many branches must be done to remove the symmetry
- Recognizing symmetry and designing algorithms to exploit the symmetry are fundamentally important

Grid Programmers Do It In Parallel



- Nodes in disjoint subtrees can be evaluated independently
- But this is not a ~~embarrassingly~~ pleasantly parallel operation
- We use the master-worker parallelization scheme

Use Master-Worker!



- Master:
  - Send task (node) to workers
- Worker:
  - Evaluate node and send result to master

COR@L
COMPUTATIONAL OPTIMIZATION RESEARCH AT LEHIGH

MW

- Master-Worker is a flexible, powerful framework for Grid Computing
- It's easy to be fault tolerant
- It's easy to take advantage of machines whenever they are available
- You can be flexible and adaptive in your approach to computing
- Everyone can have access to a powerful computing platform

MW—We're Here to Help!

- MW is a C++ software package that encapsulates the abstractions of the Master-Worker paradigm
- Allows users to easily build master-worker type computations running on Condor-provided computational grids
- It's Free, Like Free Beer: `http://www.cs.wisc.edu/condor/mw`

MW Classes

- MWMaster
  - `get_userinfo()`
  - `setup_initial_tasks()`
  - `pack_worker_init_data()`
  - `act_on_completed_task()`
- MWTask
  - `(un)pack_work`
  - `(un)pack_result`
- MWWorker
  - `unpack_worker_init_data()`
  - `execute_task()`

We're Here To Help!

Please contact Greg or Myself if you need help getting set-up with MW

Condor + MW — Cobbling Together Resources

1. Condor Flocking
   - Jobs submit to local pool run in remote pools
2. Condor Glide-in (or manual “hobble-in”)
   - Batch-scheduled resources join existing Condor pool.
3. Remote Submit
   - Log-in and submit worker executables remotely
   - Can use port-forwarding for hard-to-reach private networks

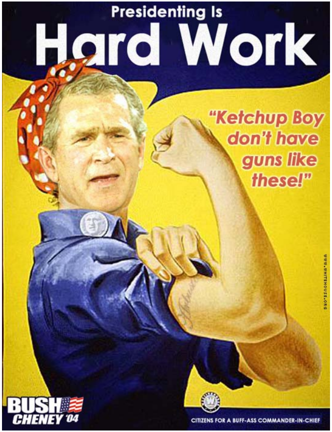
---

Schedd-on-the-side

- A new Condor technology which takes idle jobs out of the local Condor queue, translates them into Grid jobs, and uses Condor-G to submit them to a remote Grid queue
- Perfect for OSG!

Deja Vu

- In 2006, we had almost estalished a lower bound of $70 \leq |C_6|^* \leq 73$



Statistics so far...

Wall Time	47.15 days
CPU Time	57.31 years
Avg Workers	467.3
Max Workers	1253
Total Nodes	8.37×10^8
Total LP Pivots	6.53×10^{11}
Parallel Performance	94.9%

COR@L
COMPUTATIONAL OPTIMIZATION RESEARCH AT LEHIGH

"Mission Accomplished"

- We were able to establish a lower bound of $70 \leq |C_6|^* \leq 73$

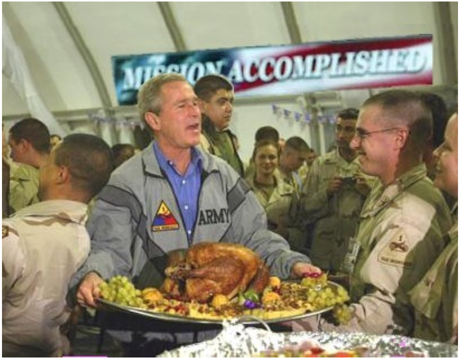
Computational Statistics

Wall Time	72.3 days
CPU Time	110.1 years
Avg Workers	555.8
Max Workers	2038
# Different Workers	4266
Total Nodes	2.85×10^9
Total LP Pivots	2.65×10^{12}
Parallel Performance	90.3%

COR@L
COMPUTATIONAL OPTIMIZATION RESEARCH AT LEHIGH

Our Continuing Mission

- I couldn't think of a good analogy, but it has been 312 days since we declared "mission accomplished": $70 \leq |C_6|^* \leq 73$



But We Press On

- Can we improve the lower bound even more?
- We've ramped up the scale of resources available

COR@L
COMPUTATIONAL OPTIMIZATION RESEARCH AT LEHIGH

Resources Used in Computation

Site	Access Method	Arch/OS	Machines
Wisconsin - CS	Flocking	x86_32/Linux	975
Wisconsin - CS	Flocking	Windows	126
Wisconsin - CAE	Remote submit	x86_32/Linux	89
Wisconsin - CAE	Remote submit	Windows	936
Lehigh - COR@L Lab	Flocking	x86_32/Linux	57
Lehigh - Campus	Remote Submit	Windows	803
Lehigh - Beowulf	ssh + Remote Submit	x86_32	184
Lehigh - Beowulf	ssh + Remote Submit	x86_64	120
TG - NCSA	Flocking	x86_32/Linux	494
TG - NCSA	Flocking	x86_64/Linux	406
TG - NCSA	Hobble-in	ia64-linux	1732
TG - ANL/UC	Hobble-in	ia-32/Linux	192
TG - ANL/UC	Hobble-in	ia-64/Linux	128
TG - TACC	Hobble-in	x86_64/Linux	5100
TG - SDSC	Hobble-in	ia-64/Linux	524
TG - Purdue	Remote Submit	x86_32/Linux	1099
TG - Purdue	Remote Submit	x86_64/Linux	1529
TG - Purdue	Remote Submit	Windows	1460

OSG Resources Used in Computation

Site	Access Method	Arch/OS	Machines
OSG - Wisconsin	Schedd-on-side	x86_32/Linux	1000
OSG - Nebraska	Schedd-on-side	x86_32/Linux	200
OSG - Caltech	Schedd-on-side	x86_32/Linux	500
OSG - Arkansas	Schedd-on-side	x86_32/Linux	8
OSG - BNL	Schedd-on-side	x86_32/Linux	250
OSG - MIT	Schedd-on-side	x86_32/Linux	200
OSG - Purdue	Schedd-on-side	x86_32/Linux	500
OSG - Florida	Schedd-on-side	x86_32/Linux	100
		OSG:	**2758**
		Total:	19,012

COR@L COMPUTATIONAL OPTIMIZATION RESEARCH AT LEHIGH

Mission Accomplished-er

- We have been able to establish $71 \leq |C_6^*| \leq 73$

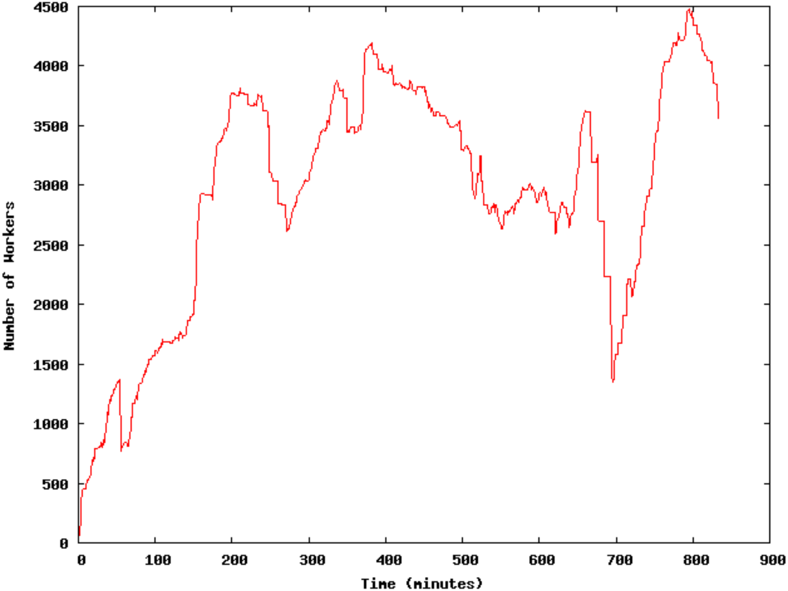
Computational Statistics

Avg. Workers	562.4
Max Workers	1775
Worker Time (years)	30.3
Wall Time (days)	19.7
Nodes	1.89×10^8
LP Pivots	1.82×10^{11}

"Mission Accomplished-est": Working on $72 \leq |C_6^*| \leq 73$

- Brings the total to $>$ 200 CPU Years!

M = 71, Number of Processors (Slice)



Working Paper

Greatest Title Ever!

- With thanks to Greg Thain for the title (among other things)

  "The Tera-Gridiron," A "Natural-Turf" for High-Throughput Computing.

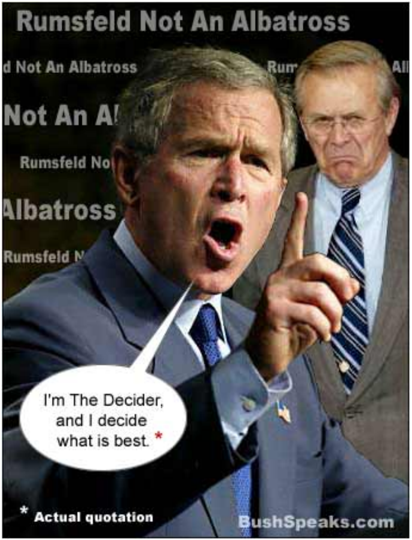
- Submit to Teragrid '07

Rejected

- Perhaps the program committee wishes to wait until the problem is finally solved.
- I have one thing to say to them...

COR@L COMPUTATIONAL OPTIMIZATION RESEARCH AT LEHIGH

You Can't Legislate a Timeline for Completion!

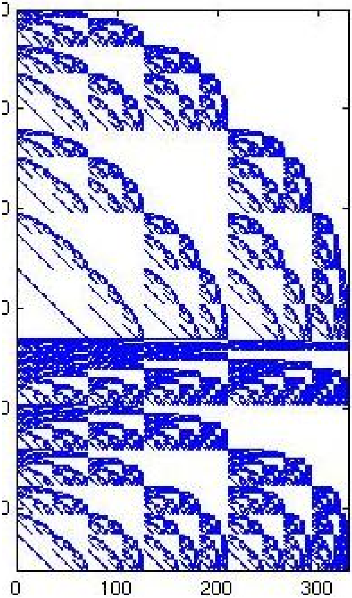


Stupid Tax-o-crats!

- Having my fragile ego crushed, we may really declare "mission accomplished" and turn to the solution of other "important", pretty integer programs
- Our approach this time is making use of new `MWBlackBox` Framework

Pretty IPs

- Steiner Triples
- Covering Designs
- Error Correcting Codes



`MWBlackBox` = Dynamic DAGMAN

- Each of the nodal subproblems is in-fact a smaller version of the original problem (with some variables fixed/removed).
- Can we use the exact same (blackbox) software that is used to solve the full-scale problem?
- Have user write no code for Task or Worker classes.
- Many people would like this functionality. Maybe you would too.

Condor DAGMAN

- Designed for static job dependencies
- Simple, Robust, Reliable
- Have to pay Condor executable startup overhead

MW Blackbox

- Designed for dynamic job dependencies
- C++ coding must be done. Less battle-tested
- Amortizes Condor executable startup overhead

BlackBox Snippets

```
MWTask_blackbox(const list<string> &args,
                const list<string> &input_files,
                const list<string> &output_files);
```

Creating Blackbox Tasks

1. List of arguments to the executable
2. List of input files (shipped to the worker executing the task)
3. List of output files: automatically shipped back to the master

MWBlackBox Snippets

```
BlackboxIPMaster::get_userinfo(int argc, char *argv[])
{
    // Set target number of workers
  RMC->set_target_num_workers(512);
    // Set name of blackbox executable
  set_executable(string("cplex"));
    // Can also stage files on the workers
  add_staged_file(param_.getProblemName());
}
```

More MWBlackBox Snippets

```
BlackboxIPMaster::act_on_completed_task(MWBlackboxTask *bbt)
{
    // Can get file streams for standard output
  ifstream stdoutfile = bbt->getStdoutStream();
    // Or open streams from output file
  string ofname = bbt->getOutputFiles()[0];
  ifstream ofile =  ifstream(ofname.c_str());
    // Then must parse output stream.
    //  1) Collect statistics or
    //  2) Create new tasks
}
```

Conclusions

- The Football Pool Problem is hard!, but now $71 \leq |C_6^*| \leq 73$
- The burning question: Will Miron let me speak about $72 \leq |C_6^*| \leq 73$ next year?
- Most important: Real large-scale applications can take advantages of all the computing power that's out there – Local grids, Tera-grids, Open-Science Grids
- MW (via Condor) can help pull all of this together
- `MWBlackBox`: Write no code for worker and task. Why not try it out?

COR@L COMPUTATIONAL OPTIMIZATION RESEARCH AT LEHIGH

Any Questions

- www.cs.wisc.edu/condor/mw
- Please talk to Greg or I
- We'd be happy to help you get started with MW
- mailto:
  - jtl3@lehigh.edu
  - gthain@cs.wisc.edu
  - mw@cs.wisc.edu