The Football Pool Problem

Jeff Linderoth

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

OSG Consortium All Hands Meeting
San Diego Supercomputing Center
San Diego, CA
March 5, 2007

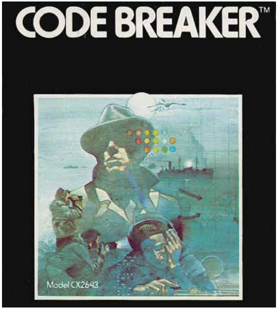
Thanks! NSF OCI-0330607, DOE DE-FG02-05ER25694 for support

Collaborators

$\left\{\right.$ FRANÇOIS MARGOT
Carnegie Mellon

$\left\{\right.$ GREG THAIN
UW-Madison

Motivation—Code Design



- W : Set of all "words" of length ν from alphabet $\{0, 1, \ldots \alpha - 1\}$. ($|W| = \alpha^\nu$)
- A code is a subset $C \subseteq W$
- Hamming distance of two words: $a \in W, b \in W$, $\text{dist}(a, b) = |\{i \mid a_i \neq b_i\}|$

Codes Appear Lots of Places

- Statistics
- Computational Biology
- Cryptography
- Computer Hardware Design

COR@L
COMPUTATIONAL OPTIMIZATION RESEARCH AT LEHIGH

Code Applications

Communications: Error Correcting Code

- Find a subset of words that are all "far apart"
- $C \subset W$ such that $a \in C, b \in C \Rightarrow \text{dist}(a, b) \geq 2d + 1$
- Maximize $|C|$
- Application: Words in C submit over a "noisy" channel on which at most d letters are changed can be "self-corrected."

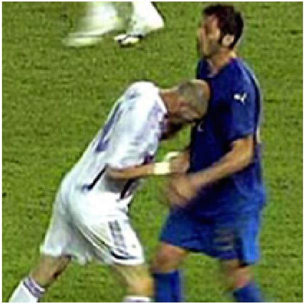
Covering Code

- Find a subset of words that "covers" the original words. (Every word $w \in W$ is at most a distance d away from a word $w \in C$)
- Find $C \subset W$ such that $\text{dist}(w, C) \leq d \ \forall w \in W$
- Minimize $|C|$
- Application: Something far more practical

Are You Ready for Some Football?!

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

ν	1	2	3	4	5
$\lvert C_\nu^*\rvert$	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 only known that
$$65 \leq C_6^* \leq 73$$

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

$$\min e^T x$$

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

Solving IPs in a Nutshell

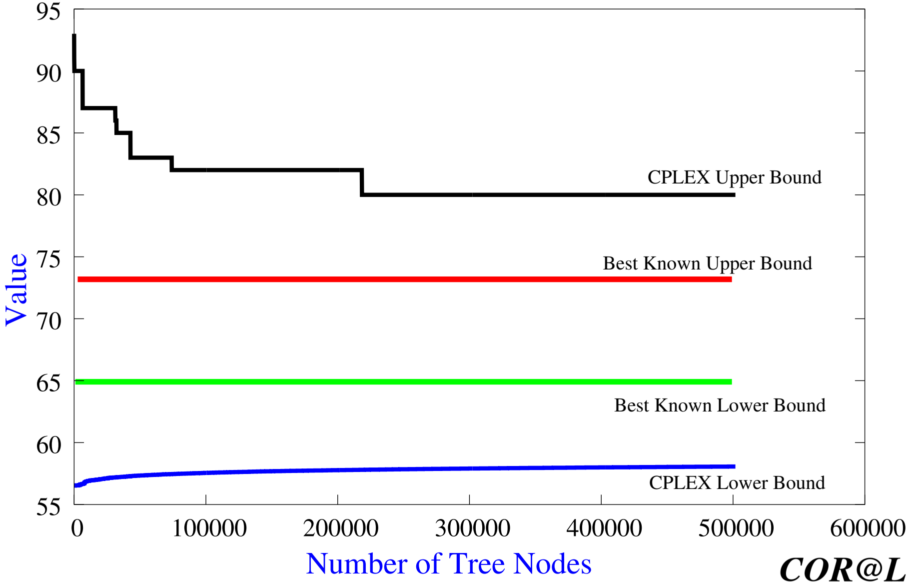
- Problem is in general $\mathcal{N}$P-Hard
- Loads of theory and techniques going back > 40 years
- Workhorse algorithm is a tree-search procedure known as branch-and-bound.
- But really, branch-and-bound or its souped-up cousin branch-and-cut have been replaced in the most part by the new technique: give-it-to-CPLEX
- CPLEX: A commercial IP package that is putting integer programmers out of business.
- CPLEX routinely solves 0-1 integer programs with thousands of variables

CPLEX Can Solve Every IP

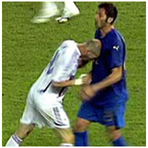
```
        Nodes                                         Cuts/
   Node  Left     Objective  IInf  Best Integer     Best Node    ItCnt     Gap

      0     0       56.0769   729                     56.0769     2200
*     0+    0                   0     243.0000        56.0769     2200   76.92%
*     0+    0                   0     110.0000        56.0769     2200   49.02%
                    56.5164   729     110.0000    Fract:  56     2542   48.62%
*     0+    0                   0     107.0000        56.5164     2542   47.18%
                    56.5279   729     107.0000     Fract:  6     2673   47.17%
*     0+    0                   0      94.0000        56.5279     2673   39.86%
*     0+    0                   0      93.0000        56.5279     2673   39.22%
Elapsed time =  90.03 sec. (tree size =  0.00 MB)
*    50+   50                   0      91.0000        56.5285    12242   37.88%
Elapsed time = 6841.16 sec. (tree size = 14.12 MB)
  31100 30002       60.1690   544      87.0000        57.1864  5467339   34.27%
  31200 30102       77.7888   216      87.0000        57.1864  5499451   34.27%
* 31200+28950                   0      86.0000        57.1864  5499451   33.50%
  31300 29044       58.9809   611      86.0000        57.1870  5511005   33.50%
Elapsed time = 9500.15 sec. (tree size = 18.70 MB)
  42700 39098       78.3242   197      85.0000        57.2845  7623200   32.61%
* 42740+36552                   0      83.0000        57.2845  7626440   30.98%
Elapsed time = 117349.90 sec. (tree size = 202.88 MB)
Nodefile size = 74.98 MB (61.52 MB after compression)
 465100 434311       66.8425   410      80.0000        58.0439 92473005   27.45%
```

NOT!


95
90
85
80
75
70
65
60
55
Value
0
100000
200000
300000
400000
500000
600000
Number of Tree Nodes
CPLEX Upper Bound
Best Known Upper Bound
Best Known Lower Bound
CPLEX Lower Bound

Plan of Attack



Apply A Hodgepodge of Tricks

1. **Isomorphism Pruning**: Trick for efficiently ordering search so that nodes that lead to symmetric solutions are not evaluated
2. **Subcode Enumeration**: Enumerate portions of potential codes of cardinality M.
3. **Subcodes and Integer Programming**: Demonstrate (via integer programming) that none of the portions of potential codes leads to a code of size M.
4. **Subcode Inequalities and Variable Aggregation**: The partial solutions can be aggregated and regrouped a bit to lessen the workload
5. **Give it massive computing power**: OSG!

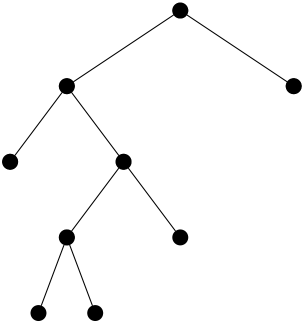
It Doesn’t Sound Like a Good Idea

- After all that hard that hard theoretical and enumerative work, we transformed 1 IP into 1000.

M	# Potential Codes
66	7
67	13
68	45
69	102
70	176
71	264
72	393
	1000

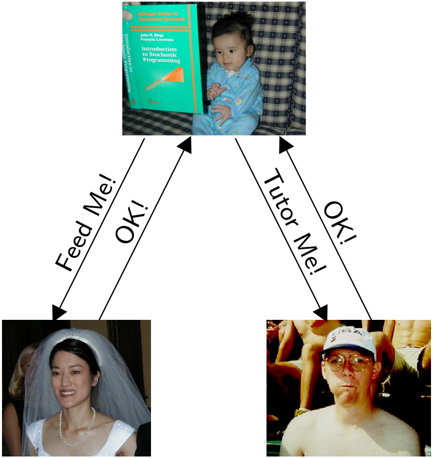
- For a given value of M, solving the related instances establishes that no code C of that cardinality exists
- We solve each of the 1000 IPs on the grid

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

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

---

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!: `http://www.cs.wisc.edu/condor/mw`

Mechanisms for Building our Grid

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!

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

Scalability for Large-Scale Computing

- Master-worker computations are perfect for such a dynamic and disperse platform
- But does it scale!?

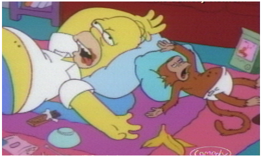
YES!—Engineer the Algorithm to the Platform!

1. Dynamic Grain Size
2. Intelligent Task Scheduling
3. Fault Tolerance (both Master and Workers)
4. Infrastructure Scaling
    - Task and network read timeouts very important
    - `epoll()` instead of `poll()`

The $64 Question

How far can it scale?

Working Hard!



Partial Computational Statistics

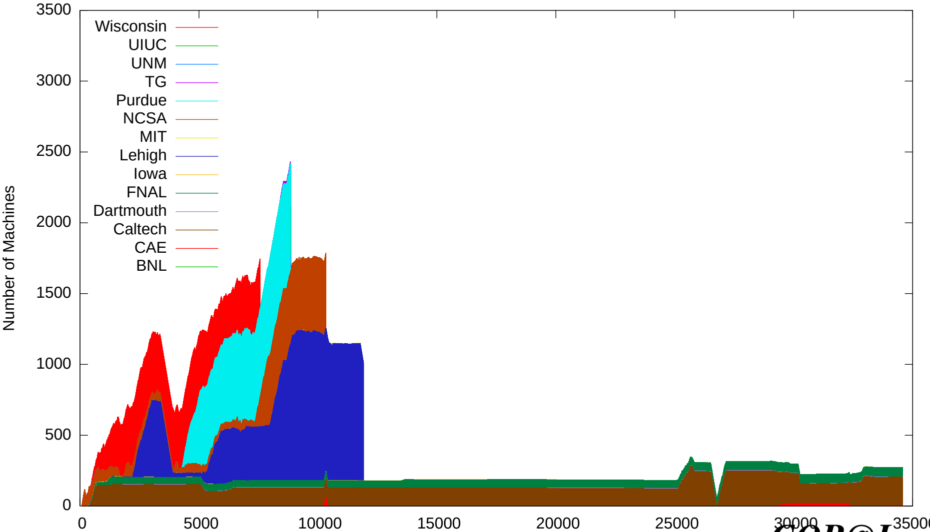
	$M = 69$	$M = 70$
Avg. Workers	555.8	562.4
Max Workers	2038	1775
Worker Time (years)	110.1	30.3
Wall Time (days)	72.3	19.7
Nodes	2.85×10^9	1.89×10^8
LP Pivots	2.65×10^{12}	1.82×10^{11}

Working on $M = 71$

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

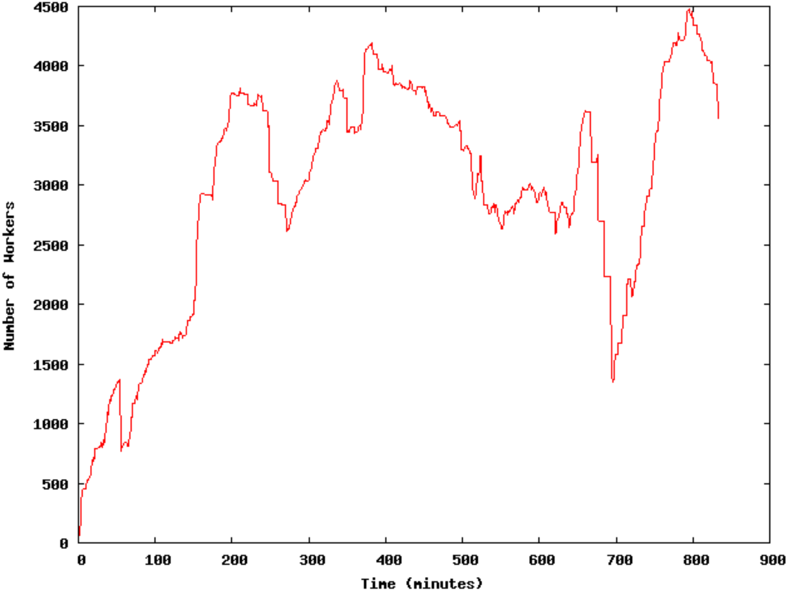
COMPUTATIONAL OPTIMIZATION RESEARCH AT LEHIGH

Computation Slice—Participating Processors



COR@L
COMPUTATIONAL OPTIMIZATION RESEARCH AT LEHIGH

$M = 71$, Number of Processors (Slice)



Conclusions

- The Football Pool Problem is hard!, but now $71 \leq |C_6^*| \leq 73$
- The Open Science Grid is available to help you with your hardest computational problems
- Being flexible and adaptive in your approach to computing can lead to significant computing power: Thank You Condor!
- We'd be happy to help you get started with MW if your computations fit into master-worker framework
- MW: `http://www.cs.wisc.edu/condor/mw`
- mailto: `jtl3@lehigh.edu`

COR@L
COMPUTATIONAL OPTIMIZATION RESEARCH AT LEHIGH