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

François Margot
Carnegie Mellon

Greg Thain
UW-Madison
Motivation—Code Design

- $W$: Set of all “words” of length $v$ from alphabet $\{0, 1, \ldots \alpha - 1\}$. ($|W| = \alpha^v$)
- 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
Code Applications

Communications: Error Correcting Code
- Find a subset of words that are all “far apart”
  \[ C \subset W \text{ such that } a \in C, b \in C \Rightarrow \text{dist}(a, b) \geq 2d + 1 \]
- 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

<table>
<thead>
<tr>
<th>$\nu$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>C^*_\nu</td>
<td>$</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>

### The Football Pool Problem

What is $|C^*_6|$?

- 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**

$$
\begin{align*}
\min & \quad e^T x \\
\text{s.t.} & \quad Ax \geq e \\
& \quad x \in \{0, 1\}^{|W|}
\end{align*}
$$
Solving IPs in a Nutshell

- Problem is in general \( \mathcal{NP} \)-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

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Objective</th>
<th>IIinf</th>
<th>Best Integer</th>
<th>Cuts/</th>
<th>Best Node</th>
<th>ItCnt</th>
<th>Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>56.0769</td>
<td>729</td>
<td></td>
<td></td>
<td>56.0769</td>
<td>2200</td>
<td></td>
</tr>
<tr>
<td>*</td>
<td>0+</td>
<td>0</td>
<td>243.0000</td>
<td></td>
<td>56.0769</td>
<td>2200</td>
<td>76.92%</td>
</tr>
<tr>
<td>*</td>
<td>0+</td>
<td>0</td>
<td>110.0000</td>
<td></td>
<td>56.0769</td>
<td>2200</td>
<td>49.02%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>56.5164</td>
<td>729</td>
<td>110.0000</td>
<td>2542</td>
<td>48.62%</td>
</tr>
<tr>
<td>*</td>
<td>0+</td>
<td>0</td>
<td>107.0000</td>
<td></td>
<td>56.5164</td>
<td>2542</td>
<td>47.18%</td>
</tr>
<tr>
<td>*</td>
<td>0+</td>
<td>0</td>
<td>94.0000</td>
<td></td>
<td>56.5279</td>
<td>2673</td>
<td>39.86%</td>
</tr>
<tr>
<td>*</td>
<td>0+</td>
<td>0</td>
<td>93.0000</td>
<td></td>
<td>56.5279</td>
<td>2673</td>
<td>39.22%</td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>Node Left</th>
<th>Objective</th>
<th>IIinf</th>
<th>Best Integer</th>
<th>Cuts/</th>
<th>Best Node</th>
<th>ItCnt</th>
<th>Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>31100 30002</td>
<td>60.1690</td>
<td>544</td>
<td>87.0000</td>
<td></td>
<td>57.1864</td>
<td>5467339</td>
<td>34.27%</td>
</tr>
<tr>
<td>31200 30102</td>
<td>77.7888</td>
<td>216</td>
<td>87.0000</td>
<td></td>
<td>57.1864</td>
<td>5499451</td>
<td>34.27%</td>
</tr>
<tr>
<td>* 31200+28950</td>
<td>0</td>
<td>86.0000</td>
<td>57.1864</td>
<td>5499451</td>
<td>33.50%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>31300 29044</td>
<td>58.9809</td>
<td>611</td>
<td>86.0000</td>
<td></td>
<td>57.1870</td>
<td>5511005</td>
<td>33.50%</td>
</tr>
</tbody>
</table>

Elapsed time = 9500.15 sec. (tree size = 18.70 MB)

<table>
<thead>
<tr>
<th>Elapsed time = 9500.15 sec. (tree size = 18.70 MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>42700 39098</td>
</tr>
<tr>
<td>* 42740+36552</td>
</tr>
</tbody>
</table>

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%|

Jeff Linderoth  (Lehigh University)
NOT!

The Football Pool Problem
OSG All-Hands Meeting 10 / 26
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 theoretical and enumerative work, we transformed 1 IP into 1000.

<table>
<thead>
<tr>
<th>M</th>
<th># Potential Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>66</td>
<td>7</td>
</tr>
<tr>
<td>67</td>
<td>13</td>
</tr>
<tr>
<td>68</td>
<td>45</td>
</tr>
<tr>
<td>69</td>
<td>102</td>
</tr>
<tr>
<td>70</td>
<td>176</td>
</tr>
<tr>
<td>71</td>
<td>264</td>
</tr>
<tr>
<td>72</td>
<td>393</td>
</tr>
<tr>
<td></td>
<td>1000</td>
</tr>
</tbody>
</table>

- 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
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

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

<table>
<thead>
<tr>
<th>Site</th>
<th>Access Method</th>
<th>Arch/OS</th>
<th>Machines</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSG - Wisconsin</td>
<td>Schedd-on-side</td>
<td>x86_32/Linux</td>
<td>1000</td>
</tr>
<tr>
<td>OSG - Nebraska</td>
<td>Schedd-on-side</td>
<td>x86_32/Linux</td>
<td>200</td>
</tr>
<tr>
<td>OSG - Caltech</td>
<td>Schedd-on-side</td>
<td>x86_32/Linux</td>
<td>500</td>
</tr>
<tr>
<td>OSG - Arkansas</td>
<td>Schedd-on-side</td>
<td>x86_32/Linux</td>
<td>8</td>
</tr>
<tr>
<td>OSG - BNL</td>
<td>Schedd-on-side</td>
<td>x86_32/Linux</td>
<td>250</td>
</tr>
<tr>
<td>OSG - MIT</td>
<td>Schedd-on-side</td>
<td>x86_32/Linux</td>
<td>200</td>
</tr>
<tr>
<td>OSG - Purdue</td>
<td>Schedd-on-side</td>
<td>x86_32/Linux</td>
<td>500</td>
</tr>
<tr>
<td>OSG - Florida</td>
<td>Schedd-on-side</td>
<td>x86_32/Linux</td>
<td>100</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>Total</strong></td>
<td><strong>19,012</strong></td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th></th>
<th>$M = 69$</th>
<th>$M = 70$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. Workers</td>
<td>555.8</td>
<td>562.4</td>
</tr>
<tr>
<td>Max Workers</td>
<td>2038</td>
<td>1775</td>
</tr>
<tr>
<td>Worker Time (years)</td>
<td>110.1</td>
<td>30.3</td>
</tr>
<tr>
<td>Wall Time (days)</td>
<td>72.3</td>
<td>19.7</td>
</tr>
<tr>
<td>Nodes</td>
<td>$2.85 \times 10^9$</td>
<td>$1.89 \times 10^8$</td>
</tr>
<tr>
<td>LP Pivots</td>
<td>$2.65 \times 10^{12}$</td>
<td>$1.82 \times 10^{11}$</td>
</tr>
</tbody>
</table>

**Working on $M = 71$**

- Brings the total to $> 200$ CPU Years!
Computation Slice—Participating Processors

Number of Machines vs Time for various institutions:
- Wisconsin
- UIUC
- UNM
- TG
- Purdue
- NCSA
- MIT
- Lehigh
- Iowa
- FNAL
- Dartmouth
- Caltech
- CAE
- BNL

Time points:
- 0
- 500
- 1000
- 1500
- 2000
- 2500
- 3000
- 3500

Number of Machines:
- 0
- 500
- 1000
- 1500
- 2000
- 2500
- 3000
- 3500

Legend:
- Wisconsin
- UIUC
- UNM
- TG
- Purdue
- NCSA
- MIT
- Lehigh
- Iowa
- FNAL
- Dartmouth
- Caltech
- CAE
- BNL
$M = 71$, Number of Processors (Slice)
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](http://www.cs.wisc.edu/condor/mw)

mailto: jtl3@lehigh.edu