A Distributed Search for Strongly Regular Graphs

Derrick Stolee
(\text{with Stephen G. Hartke})

Spring 2009
Outline

1. Motivation
2. Tools
3. Software
4. Progress
5. Future
Motivation: Strongly Regular Graphs

- Interesting discrete objects.
- Four parameters: \((n, d, \lambda, \mu)\).
- Constructions for certain parameters.
- Proofs for impossible parameters.
- Some fall through the gaps.
- Example: \((99, 14, 1, 2)\).
Motivation: Searching for SR-Graphs

- **Standard method:** use ILP formulation.
  - TOO BIG!
  - 99 vertex requires $10^6$ variables, $10^7$ constraints.
- **Our method:** search through possible subgraphs.
  - Best for parameters with small $\lambda, \mu$.
  - Uses symmetry to remove redundancy.
  - Embarassingly parallel.
Tools

- Initial Guaranteed Structure
- Isomorph-free Generation (a/k/a Canonical Deletion)
- Constraint Satisfaction (Local)
- LP Relaxation (Global)
Cluster Software

- TreeSearch: generalized tree search management.
- Job management scripts.
- Remote monitoring of Condor queue.
Isomorph-free generation is used a LOT.

The structure of a search is almost always the same:
- Compute symmetries.
- Add a vertex.
- Compute symmetries.
- Prune?

To write parallel code each time would be a waste.
TreeSearch: Basic Idea

- Each child of a search node is completely independent.
- Each search node is described by the path from the root.
- This path is a sequence of child choices.

<table>
<thead>
<tr>
<th>Type</th>
<th>Base</th>
<th>Current</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>3</td>
<td>5</td>
<td>10</td>
<td>14</td>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>
There are four types of output:

- **Solution**: found something at this search path.
- **Complete**: the input search node has been completely explored.
- **Partial**: a timeout was reached during search, here is current position.
- **Statistics**: here are some statistics gathered.
Job Management

- Condor requires a different file for each job.
- Batches were made with 50, 500, or 1500 jobs.
- `expandjobs.py` split one file of many jobs into many input files of one job each.
- `compactjobs.py` took output files and collected partial jobs, solutions, and statistics.
- `condorsubmit.sub` or `gridsubmit.sub` were modified to fit the number of jobs found.
Remote Monitoring of Condor Queue

Big job 30139 no longer in queue
5:35 AM Nov 26th from TwitterMail

Condor update: 0 running and 0 idle jobs.
5:35 AM Nov 26th from TwitterMail

Condor update: 1 running and 0 idle jobs.
11:35 PM Nov 25th from TwitterMail

Condor update: 8 running and 0 idle jobs.
12:05 PM Nov 25th from TwitterMail

Condor update: 26 running and 0 idle jobs.
11:35 AM Nov 25th from TwitterMail
Progress

- Initial implementation generated to depth 3 created about 500 jobs.
- It became clear this would take a while.
- New mathematics brought a new initial choice and a new implementation.
- Search starts with bigger example, but less symmetry.
- Depth 2 created about 1500 jobs.
Progress: Running on Prairiefire

- Ran batches of size 50, 500, 1500 for 12 hours per job.
- A few pruned quickly.
- Most thrashed at depth 6 or 7.

```
T MAX DEPTH  7.0
T SUM TOTAL_TIME 24654509.0
T SUM NODE_COUNT  7357944.0
T SUM PRESOLVE_PRUNE  6845447.0
T SUM PRUNE_COUNT  405417.0
T SUM PRESOLVE_COUNT  6900138.0
T SUM PRESOLVE_TIME 28581089.8
T SUM PRUNE_TIME   10551.07
T SUM NAUTY_TIME   19331.58
T SUM NAUTY_COUNT  471870823.0
```
Future Work

This research is incomplete!

- Recently obtained Open Science Grid credentials.
- Will generate a higher amount of jobs, run on Firefly or Red.
- Hope to push to other sites.

The search itself may be improved.

- Customize the first choice further.
- Test on smaller examples where result is known.
Lessons Learned

- Don’t submit jobs that run for hours without checking for memory leaks.
- Not all software is as advertised.
  - Condor checkpointing?
- Lots of HARDWORKING people behind these machines.
Summary

- Strongly regular graphs are fun and challenging.
- Lots of high-powered software and machines tackling the problem.
- Software built and tested may require more power (more brains?).
- Testing on smaller examples may present opportunities.