Saturation User Guide

Version 1.0

Derrick Stolee
University of Nebraska-Lincoln
`s-dstolee1@math.unl.edu`

February 27, 2012

Abstract

The Saturation software is an implementation of orbital branching with a custom augmentation to search for uniquely K_r-saturated graphs.

1 Acquiring *Saturation*

The latest version of *Saturation* and its documentation is available online as part of the *SearchLib* collection at the address

`http://www.math.unl.edu/~s-dstolee1/SearchLib/`

Saturation is made available open-source under the GPL 3.0 license.

To compile *Saturation*, use a terminal to access the `Saturation/src/` folder and type `make`. The executables will be placed in `Saturation/bin/`

1.1 Acquiring Necessary Libraries

There are two external libraries and two *SearchLib* projects used by *Saturation*.

1. *nauty* performs isomorphism and automorphism calculations. *nauty* was written by Brendan McKay [2] and is available at

   `http://cs.anu.edu.au/~bdm/nauty/`

2. *cliquer* performs clique calculations, including finding the clique number and counting the number of cliques. *cliquer* was written by Niskanen and Östergård [3] and is available at

   `http://users.tkk.fi/pat/cliquer.html`

3. *TreeSearch* is a project in *SearchLib* that abstracts the structure of a backtrack search in order to allow for parallelization. *TreeSearch* is available on the same web site as *Saturation*. Consult the *TreeSearch* documentation for details about the arguments and execution processes.

4. *Utilities* is a project in *SearchLib* containing useful objects and functions necessary by other projects in *SearchLib*. *Utilities* is available on the same web site as *Saturation*.

1.2 Full Directory Structure

For proper compilation, place the different dependencies in the following directory structure:

- `SearchLib/` – The *SearchLib* collection.
  - `Saturation/` – The *Saturation* project.
    - `bin/` – The final binaries are placed here.
    - `docs/` – This folder contains documentation.
    - `src/` – Contains source code. Compilation occurs here.
  - `TreeSearch/` – A support project from *SearchLib*.
  - `Utilities/` – A support project from *SearchLib*.
    - `src/` – Type `make` in this directory to compile the *Utilities* project.
  - `cliquer/` – The *cliquer* library must be placed and compiled here.
  - `nauty/` – The *nauty* library must be placed and compiled here.

2 Execution

There are two executables in the *Saturation* project.

- `saturation.exe` runs an orbital branching search for uniquely K_r-saturated graphs of a given order n.

- `cayley.exe` generates Cayley complements and checks if they are uniquely K_r-saturated for some r.

2.1 `saturation.exe`

This executable generates all uniquely K_r-saturated graphs of a given order n. It uses a customized orbital branching approach.

```
saturation.exe [TreeSearch args] -N # -r # [--cliquer]
```

- `-N #` specifies the number n of vertices to use. All uniquely K_r-saturated graphs of order n will be generated.

- `-r #` specifies the value of r to use when searching for uniquely K_r-saturated graphs.

- `--cliquer` is an option that specifies to use the *cliquer* library in the pruning steps of the search. If not specified, the search uses a tabulation method.

2.2 `cayley.exe`

This executable generates Cayley complements and checks if they are uniquely K_r-saturated for some r. For a fixed number of generators g, it selects a set $S = \{1 < s_2 < s_3 < \cdots < s_g\}$ and then selects integers n so that $2s_g + 1 \leq n \leq N_{\max}$. Then, it uses

To execute `cayley.exe`, use the following format of arguments:

```
cayley.exe [TreeSearch args] -N # -G # -t # [--verbose] [--dihedral]
```

- `-N #` specifies $N_{\max}$, the maximum value of n to use when searching for a uniquely K_r-saturated Cayley complement $\overline{C}(\mathbb{Z}_n, S)$.

- `-G #` specifies the number of generators to place in the set S.

- `-t #` specifies the number of seconds to allow a call to the *cliquer* library run before terminating. If a call is terminated early, the graph that was being tested is output as a job (using *TreeSearch* job descriptions).

- `--verbose` is an option to output the status of the search while testing a specific Cayley complement. Not recommended for a large-scale search, but only for a long test of a specific example.

- `--dihedral` is an option that checks for uniquely K_r-saturated Cayley complements over the dihedral groups. (*Note:* We have not yet found any generator sets that create uniquely K_r-saturated Cayley complements of dihedral groups.)

3 *TreeSearch* Arguments

- `-k #` — The killtime: How many seconds before halting the process and reporting a partial job.

- `-m #` — The maximum depth: the maximum number of steps to go before halting (or in generation mode, a new job is written at this depth).

- `run` — Run mode: The input jobs are run until finished or the killtime is reached.

- `generate` — Generation mode: The input jobs are run and new jobs are listed when reaching the maximum depth.

- `--maxjobs #` — The maximum number of jobs to generate before halting with a partial job (default: 1000).

- `--maxsols #` — The maximum number of solutions to output before halting with a partial job (default: 100).

References

[1] S. G. Hartke, D. Stolee, Uniquely K_r-Saturated Graphs, preprint (2012).

[2] B. D. McKay, nauty User's Guide (v. 2.4), Dept. Computer Science, Austral. Nat. Univ. (2006).

[3] S. Niskanen, P. R. J. Östergård, *Cliquer* user's guide, version 1.0. *Technical Report* T48, Communications Laboratory, Helsinki University of Technology, Espoo, Finland (2003).

[4] D. Stolee, TreeSearch user guide, available at http://www.github.com/derrickstolee/TreeSearch/ 2011.