Cameron-Liebler Set Enumeration Manual

Daniel Palmarin

October 21, 2020

Contents

1 SOFTWARE INSTALLATION

1.1 Programming Languages

The following software is required:

- GAP [1]
- Gurobi [2]
- Python 3 [3]

The operating system (OS) that was used in the development of this program was Windows 10. However, it has been tested successfully on Ubuntu 18.04 and macOS Catalina. The folder locations and syntax that each OS uses is slightly different. Windows 10 syntax is used in this document.

1.2 GAP Installation

Windows installation: Use the .exe installer which contains 64-bit binaries for GAP (compiled with the support of GMP and readline libraries) and for selected GAP packages. Install it to its default location.

Linux and macOS installation: It is highly suggested to install GAP using the source distribution. See the GAP installation page for details.

1.3 Running and Customizing GAP

Adding GAP to PATH: The installation of GAP on Windows comes with two .bat files: gap.bat and gapcmd.bat. We are interested in adding the latter to Windows PATH. For convenience, rename gap.bat to gap64.bat (this will be unused) and rename gapcmd.bat to gap.bat. Then follow these steps:

1. Open the Start Search, type in "env", and choose "Edit the system environment variables"
2. Click the "Environment Variables..." button.
3. Under the "System Variables" section (the lower half), find the row with "Path" in the first column, and click edit.
4. The "Edit environment variable" UI will appear. Here, you can click "New" and type in C:\gap-4.11.0\bin. Press "OK" and GAP is added to PATH.

For Linux or macOS users, GAP can be added to PATH directly from the terminal. Try adding `export PATH=\$PATH:</path/to/file>` to your /.bashrc file.

Running GAP: With GAP added to PATH, open a new PowerShell/Terminal window and type: `gap`. This will launch GAP. It is highly suggested to allow GAP access to all physical RAM installed. For example, if your machine has 64 GB of RAM, open GAP by executing the following command: `gap -o 64g`. I also like to suppress the GAP banner from startup using: `gap -b -o 64g`

Pre-loading GAP packages: The GRAPE and SONATA packages are required. It is suggested to have them automatically loaded on each GAP startup. To do this, start GAP and execute the following commands:

```
gap> pref := UserPreference( "PackagesToLoad" );;
gap> Add( pref, "grape" );;
gap> Add( pref, "sonata" );;
gap> SetUserPreference( "PackagesToLoad", pref );;
gap> WriteGapIniFile();;
```

1.4 Gurobi and Python Installation

Follow the installation steps provided by both distributors. For Gurobi, if prompted, specify Python 3 has your desired application programming interface (API). Familiarize yourself with Gurobi in Python. See the Gurobi Manual. I'd suggest going through some examples in the example tour. This will confirm that your distribution of Gurobi is functioning properly.

2 RUNNING CAMERON-LIEBLER SET ENUMERATION

2.1 Folder layout

The entire program is contained in the downloadable `CLsetEnumeration.zip`. Extract it and save its contents to an easily accessible location. For example, mine is saved to the following location in Windows: `C:\Users\Daniel\Documents\Graduate_Research\CLsetEnumeration`. Inside this folder, there should be the following folders:

- `Gurobi_logs`: Gurobi logs every computation it performs. Such logs are stored in this folder.
- `Gurobi_LP_Models`: The program will automatically generate LP models. They are stored here.
- `LaTeX_Output`: All program output is converted to a .tex file to view.
- `Python_Scripts`: The file `MAIN.g` will execute each of these python scripts.
- `Solutions_Summary_Files`: All program output is reported to a .txt file. This will also include all CL sets found (in list format).
- `Temp_Textfiles`: `MAIN.g` and `GurobiReader.py` share variables through .txt files. They are created and removed in this folder. They should not be edited or deleted during the execution of `MAIN.g`.

2.2 Initializing the program

Open `MAIN.g`. On Windows, I use Notepad++ to edit all of my .g and .py files. I use the .py language theme for both. On Ubuntu and macOS, I use Atom to edit all of my .g and .py files.

- Under `SECTION 0 - PREAMBLE`, find the variable `home_dir`. Set this to the location where you saved the program folder. Also, set the `python3` variable to the location of python on your device. For example, for me these are both set to:

  ```
  home_dir := "C:/Users/Daniel/Documents/Graduate_Research/CLsetEnumeration/";
  python3 := "C:/Users/Daniel/AppData/Local/Programs/Python/Python37/python.exe";
  ```

2.3 Running the program

Open a new PowerShell/Terminal window and start a new GAP session. The `MAIN.g` is run by using the `Read` command. For me, I use:

```
gap> Read("C:/Users/Daniel/Documents/Graduate_Research/CLsetEnumeration/MAIN.g");
```

There are various options that can be adjusted:

- `grp := PrimitiveGroup(n,k)`

  You can input whichever group you would like to analyze however you would like, as long as GAP recognizes it as a group. I prefer to use the `PrimitiveGroups` library. You'll find a list of all 2-transitive groups from degrees 1-30 in the file `2TransGroups.pdf`. The primitive group GAP index is included for each group. Adjust the `group_name` variable as you see fit.

- `starting_LP` and `ending_LP`

  This specifies which linear programming (LP) models to analyze using Gurobi. To find all Cameron-Liebler sets, `starting_LP` should be equal to 1 and `ending_LP` should be equal to `Int(n/2)`. If you only wish to find maximum cocliques, `ending_LP` should be equal to 1.

- `cont_program`

  This can be equal to either 0 or 1. If equal to 0, then all new .lp, .tex, .txt, and .log files will be produced. If `group_name` is not changed between separate program executions, the old files will be overwritten. Setting `cont_program` to 1 should only be used if you had to abort the computation for some reason, but now you wish to continue without having to remake the LP files. This assumes that you've run the program with `cont_program := 0` at least once so that all files required have been already generated.

- `clique_type`

  This controls the number of maximum cliques that will be generated. The options are as follows:

  - 0: This is the default. The program determines if the maximum cliques that are subgroups span a space of dimension $|G| - (n-1)^2$. If yes, it uses option 1 below; if no, it uses option 2.
  - 1: Only maximum cliques that are subgroups (and their cosets) are generated. This uses the `SONATA` package, which is preloaded (see Section 1.3). Note: using this option may cause errors.
  - 2: All maximum cliques are generated. This uses the `GRAPE` package, which is preloaded (see Section 1.3). Note: using this option may cause errors.

- `ij_mappings`

  This can be equal to 0 or 1. If 0, then the canonical cocliques (the stabilizers and their cosets) are *not* constrained to the LP models initially (Gurobi will find them as a solution). If equal to 1, then the canonical cocliques *are* constrained to the LP models. Note that Gurobi may return no solutions if this option is used (which implies that the only Cameron-Liebler sets are the canonical cocliques). If the group in question is 2-transitive and has a very large order, I'd suggest using option 1.

- `extra_tests`

  This is a list that can contain the numbers $1-4$. This determines various group and maximum coclique properties by calling `Analyses.g`. You can specify which (if any) tests you would like to perform on the maximum cocliques found. Note that these tests are performed after Gurobi finds all solutions in the first LP file ($c = 1$). After completing these tests, the program will move to finding solutions in the remaining LP files specified ($2 <= c <= \lfloor \frac{n}{2} \rfloor$). The tests performed are as follows (see Section 4.1 in my thesis for more details):

  1. group degree, order, transitivity, Frobenius or non-Frobenius, and the number of components in Γ_G;
  2. the strict EKR property and other relevant maximum coclique properties;
  3. the dimensions of $\mathcal{C}'$, $\mathcal{C}$, and W;
  4. the spectrum of Γ_G and whether or not the ratio bound holds with equality.

- `tex_output`

  If this is equal to 0 then a .tex file is generated, which contains all relevant data found by GAP and Gurobi (except the precise solutions themselves, which are stored in the solutions summary .txt file). No .tex is generated if this is equal to 1.

2.4 Gurobi options

In the `Python_Scripts` folder, open the `GurobiReader.py`. Here, you have the ability to adjust Gurobi's parameters. See the code under the `#Analyze model` section. You'll find a few useful parameters, such as:

- `TimeLimit`

  This allows you to specify a maximum computation time (in seconds). This is useful if you find that Gurobi is taking too long, or if you wish to suddenly skip an LP model(s) in the middle of a long computation. For example, suppose a computation has been running for over 20 hours. If you don't want to terminate the program manually (and lose the generation of the .tex file) you can set the `TimeLimit` to 1 second, and after Gurobi times out of its current maximum value (say 18000 seconds), it will cease computation on all further LP models after 1 s. In other words, `GurobiReader.py` can be edited during the execution of `MAIN.g` (this should be done cautiously).

- `NodeLimit`

  I have this set to 1 million nodes. If the number of nodes exceeds 1 million, you may encounter a memory error.

- `NodefileStart`

  When the amount of memory used to store nodes (measured in GBs) exceeds the specified parameter value, nodes are compressed and written to the disk. The performance penalty is usually less than 10 %. I don't use this by default. If I encounter a memory error, I start by setting this to 0.5. If I still get a memory error, I increase this to 1.

- `Threads`

  If you encounter a memory error (and your CPU has many cores), I'd suggest reducing the number of threads that Gurobi can use.

References

[1] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.11.0*, 2020.

[2] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2020.

[3] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.