Manual to functions used in Optimal regular graph designs

Sera Aylin Cakiroglu*
School of Mathematical Sciences, Queen Mary University of London,
Mile End Road, London E1 4NS, UK
Email: aylin.cakiroglu@crick.ac.uk

1 System requirements and the basic workflow

System requirements:

- a recent working version of Mathematica (`www.http://www.wolfram.com/mathematica/`); we used version 9.0 for Mac OS X x86 (64-bit) (January 24, 2013), however we have tested all function on version 10.0, the current version at the time of writing; and
- a working version of GAP (`http://www.gap-system.org/`); and
- most of the functions are designed to run either in sections or to read the files line by line if not enough RAM is available. However, we tested all functions on the largest case (`v` = 14, `degree` = 5) on a MacBookPro with 16GB RAM (16 GB 1600 MHz DDR3, Processor 2.6 GHz Intel Core i5), and did not need to use those options.

To find the best RGDs for given v, k and r:

1. download and install the program `genreg` (available for download at `http://www.mathe2.uni-bayreuth.de/markus/reggraphs.html`) and generate the adjacency lists of all graphs with the program `genreg` (option `-a`) and copy the file `[v]_[degree]_3.asc` into the working directory. Alternatively, use the provided adjacency lists in the zip directory GenregOutput.zip
2. change the first line in the attached Mathematica notebook 'OptimalRGDs.nb' to your working directory and evaluate the cell. You might want to open a new notebook (within the same kernel) to then run the following functions:
   - `GetValuesFrom[v,degree,startgraphnumber,sectioned]` to compute the D-value for every graph as a polynomial in y. This function saves the coefficients in the file "CoefficientList.txt" in the working directory.
   - `FindStabilizingValue[v,degree,maxY,Memory]` to compute the A- and D-values for all $y \leq maxY$ and to compare the orders (in regard to A- and D-value). In all our examples $maxY = \delta + 2$ was enough to find the value for y for which the orders stabilize. The function will print whether there was change in the orders (or not) on the screen .
   - `ADOrder[v, degree, stabvalue, startposition, Memory]` to verify that the value for y found in the previous step is indeed the value for which the orders stablize. It will print whether there was an order change for any y bigger than the assumed stabilization value or

*Present address: Francis Crick Institute, 1 Midland Road, London NW1 1AT, UK

if numerical methods had to be used on the screen. If there is a change in the order of the matrices in either criteria, then the function terminates with a warning.

- `CopyBestGraphs[v, degree, optimality, min, max]` to create the files `[optimality]BestGraph[rank].txt` containing the adjacency list of the graph ranked [rank] in the order for the [optimality] criterion for `min`≤[rank]≤ `max`. in the optimal order which can be read in by the following GAP functions to search for a design.

3. open the file main.g in a text editor. You will need to change the variable `pathToDir` on the first line of main.g to the absolute path of your working directory. Then open GAP and execute the command `Read("$path/main.g");` where `$path` stands for the absolute path to the file main.g. Executing this command will load all necessary packages and functions and will set the directories. To find the best RGDs for v points execute the command `GetDesigns(v,optimality,minLambda,maxLambda);`. This will generate all designs on `v` points with for all admissible block sizes with `minLambda`≤ λ ≤`maxLambda` and rank less than 100. For other cases use `TestForBlockSize(path,v,degree,optimality,max_rank,blockSize,minLambda,maxLambda)` (Choose optimality from "A" or "D"; path is the absolute path to the working directory). The function will search for a RGD with block size k and any admissible $\lambda \in$ [`minLambda`, `maxLambda`]. Note that in most cases, `maxLambda` ≥ 3 will lead to long computation times.

See below for more details on the parameters and options for the different functions.

2 Functions in Mathematica

Open the notebook `OptimalRGDs.nb` and evaluate its (only) cell. The working directory is set to `wdir = "~/Documents/OptimalRGDS/Programs and Data/"` which can be changed in the notebook on the first line. However, `wdir` should be set to the directory containing the `genreg` output file `[v]_[degree]_3.asc`. For all further computations it is recommended to open a new notebook.

GetValuesFrom[v,degree,startgraphnumber,sectioned]

`GetValuesFrom[v,degree,startgraphnumber,sectioned]` reads the file `[v]_[degree]_3.asc` created by `genreg` containing the adjacency lists of all regular connected graphs on v vertices and degree `degree`. It then computes their D-values as polynomials and saves the lists of the coefficients in the file 'CoefficientList.txt'. The directory where these files are stored is `wdir/[v]_[degree]` if `v`> 9 or `wdir/[0 v]_[degree]` if `v`< 10. If there are many graphs, the function can be run on several kernels (with different values of `startgraphnumber`) to create the lists of the coefficients of the D - values. This needs to be specified by setting `sectioned` = 1, other wise use `sectioned` = 0. If this option is being used, the function creates the file
'CoefficientList_[startgraphnumber].txt' containing the lists of the next 500,000 graphs. These lists need to be merged and stored as a complete list in a file 'CoefficientList.txt' by hand before continuing.

Parameters :

`v`: number of vertices.

`degree`: degree of graph.

`startgraphnumber`: graph number from which to start reading and computing the values. This allows in cases where there many graphs the function to be run on several kernels, otherwise choose `startgraphnumber` = 1.

`sectioned`: flag whether the values should be computed for sets of 500,000 graphs or not (1 or 0 respectively)

Input: `[v]_[degree]_3.asc`

Output: CoefficientList.txt

Example use : `GetValuesFrom[10, 6, 1]`

FindStabilizingValue[v,degree,maxY,Memory]

`FindStabilizingValue[v,degree,maxY,Memory]` finds the value for y for which the orders (regards to the A and D-value) stabilize (up to `maxY`) if it exists. The function depends on the file 'CoefficientList.txt' computed by `GetValuesFrom`.

The list of coefficients is first turned into A- and D-values (as polynomials in y) which are then evaluated for all values of y up to `maxY`, sorted and finally compared to the previous order to detect order changes.

This function reads the file 'CoefficientList.txt' into memory if the number of graphs is small enough which needs to be specified by the user by setting `Memory`= 1.

If this is not possible (`Memory`= 0), the coefficients of each graph needs to be read in individually and the D - and A - values are computed for the particular graph and then stored.

In all cases, we were able to read the list of all values into memory.

Parameters :

`v`: number of vertices.

`degree`: degree of graph.

`maxY`: the maximum value for which the orders are computed and compared.

`Memory` : flag whether the coefficients should be read into memory or not (1 or 0 respectively).

Input: CoefficientList.txt

Output: AvaluesOrdered.txt, DvaluesOrdered.txt

Example use : `FindStabilizingValue[10, 4, 3, 1]`

ADOrder[v, degree, stabvalue, startposition, Memory]

`ADOrder[v, degree, stabvalue, startposition, Memory]` checks whether orders indeed stabilize for the previously found value for y (`stabvalue`). The function depends on the files 'AvaluesOrdered.txt' and 'DvaluesOrdered.txt' created by the function `FindStabilizingValue`. The function reads the file 'CoefficientList.txt' into memory if the number of graphs is small enough; this needs to be specified by the user by setting `Memory` = 1. If this is not possible (`Memory` = 0), the coefficients of each graph are being read in individually. The function computes the A- and D-values from the coefficient lists for any two succeeding graphs in the orders 'AvaluesOrdered.txt' and 'DvaluesOrdered.txt'. The exact Mathematica function `Solve[]` is used to find any positive root of the difference of the respective polynomials (see the main manuscript for details). If this takes too long (cut-off is 300 seconds), then a warning is issued and a numerical function (`NSolve[]`) is used. If a root is found in either cases, a warning message will be printed to screen. We never had to use `NSolve[]` in any of the presented cases.

If the values for A- and D-values differ, `stabvalue` can be taken to be the bigger one since the order of the graphs will not change for a value larger than the actual stabilization value.

If the number of graphs is large, `ADOrder[v, degree, stabvalue, startposition, Memory]` can be run on several kernels on different parts of the list by choosing `startposition`> 1.

Parameters :

`v`: number of vertices.

`degree`: degree of graph.

`stabvalue`: the stabilization value to be verified.

`startposition` : position in the order from which to start reading and computing the values. In cases where there many graphs, this allows the function to be run on several kernels, otherwise choose `startposition` $= 1$.

`Memory` : flag whether the coefficients should be read into memory or not (1 or 0 respectively).

Input: CoefficientList.txt, AvaluesOrdered.txt, DvaluesOrdered.txt

Output: printed on screen.

Example use: `ADOrder[10, 4, 1, 1, 1]`

CopyBestGraphs[v, degree, optimality, min, max]

`CopyBestGraphs[v, degree, optimality, min, max]` copies the graphs of ranks in `[min,max]` for a chosen optimality criterion as adjacency lists to the file '[optimality]bestGraph[rank].txt' to the directory `wdir/[v]/[v]_[degree]/Graphs/`. These files are needed by the GAP functions to search for a design.

Parameters :

`v`: number of vertices.

`degree`: degree of graph.

`optimality`: the type of optimality, choose from `A` or `D`.

`min, max`: minimum and maximum rank for which the adjacency lists should be written to file. In the case of many graphs this function can be run on several kernels.

Input: AvaluesOrdered.txt, DvaluesOrdered.txt

Output: Files `[optimality]bestGraph[rank].txt` for `min` $\leq$ `rank` $\leq$ `max` in directory `wdir/[v]/[v]_[degree]/Graphs/`.

Example use: `CopyBestGraphs[10,4,A,1,100]`

2.1 Example: computations for $v = 12$, degree $= 5$

```
In[1]:=GetValuesFrom[12, 5, 1]

file exists

FindStabilizingValue[12, 5, 5, 1]
```

```
A-Order Change at x= 1

D-Order Change at x= 1

A-Order Change at x= 2

D-Order Change at x= 2

A-Order Change at x= 3

D-Order Change at x= 3

ADOrder[12, 5, 3, 1, 1]

Orders for stabilizing value are the same

AOrdering stays the same for x>=3

DOrdering stays the same for x>=3

CopyBestGraphs[12, 5, "A", 1, 100]

Writing to file finished.

FindStabilizingValue[12, 5, 5, 0]

A-Order Change at x= 1

D-Order Change at x= 1

A-Order Change at x= 2

D-Order Change at x= 2

A-Order Change at x= 3

D-Order Change at x= 3

ADOrder[12, 5, 3, 1, 0]

Orders for stabilizing value are the same

AOrdering stays the same for x>=3

DOrdering stays the same for x>=3
```

3 GAP functions

TestForBlockSize

`TestForBlockSize(path,v,degree,optimality,max_rank,blockSize,minLambda,maxLambda)` reads the files that contain the adjacency lists of the graphs ordered (ranked) with regard to an optimality criterion.
The function reads the files `[optimality]bestGraph[i].txt` created by the Mathematica function `CopyBestGraphs` for $0 <= i <=$`max_rank`. The function will halt when if has found a design with desired block size and lambda given by a matrix in the file. This block design will be saved as XML file `[optimality]_BestRGD_[v]_[blockSize]_[replication]_[lambda].xml` (which can be read with GAP), and aditionally as a '.txt' file with the same prefix in the directory
`~/Documents/OptimalRGDS/Programs and Data/[v]`. Further, the rank is saved in the file

`[optimality]_Rank_[v]_[blockSize]_[replication]_[lambda].txt` in the same directory. Additionally, a log file will be created for each session and saved to the directory
`~/Documents/OptimalRGDS/Programs and Data`. The directory `path` needs to be the absolute path to the directory containing the folders created by the Mathematica function `GetValuesFrom`.

Parameters :

`path`: absolute path to the working directory that contains the folders created by the Mathematica programs (i.e. `"~/Documents/OptimalRGDS/Programs and Data/"`).

`v`: number of vertices.

`degree`: degree of graph.

`optimality`: optimality criterion, `"A"` or `"D"`

`max_rank`: the maximum possible rank the files can have or is chosen as a cut-off

`blockSize`: block size of the design

`minLambda, maxLambda`: minimum and maximum value for lambda to be used in generating the design

Input: Files `[optimality]bestGraph[rank].txt` for consecutive `rank` starting at 1 in directory `wdir/[v]/[v]_[degree]/Graphs/`

Output: The best design as .txt file
`[optimality]_BestRGD_[v]_[blockSize]_[replication]_[lambda].txt` and its rank in a separate file `[optimality]_Rank_[v]_[blockSize]_[replication]_[lambda].txt`

Example use:
```
gap> TestForBlockSize("~/Documents/OptimalRGDS/Programs and Data/",10,6,"A",50,2,0,3)
```

main

Set the correct path `pathToDir` in main.g and evaluate `Read("path/main.g");` in the terminal where `pathToDir` is the path to the working directory that contains the folders created by the Mathematica functions (i.e. `"~/Documents/OptimalRGDS/Programs and Data/"`) and the files `OptimalRGDS.g`, `main.g`. This will automatically load all packages and functions. Then run the function `GetDesigns(v,optimality,minLambda,maxLambda)` to automatically get the best RGDs with `v` points.

Example use: `gap>GetDesigns(10,"A",0,1);`