New Gadget Code Description

Vadim Korotkikh

November 2017

Abstract

This document contains the medium level detail description of the python code for generating 36k Adinkras and New Gadget values. Brief intro simple paragraph at the beginning of the document.

This document write up contains the description of the software code algorithm that is used to calculate the New Gadget for the entire BC4 Coxeter group space of 36,864 Adinkras and how it is written and executed using Python 3. Specific Python version used in calculation was Python 3.5, but the code is also compatible with Python 2.7. Software wise the code builds upon earlier developments/works by the author but with changes to the code that pertains to the final gadget calculation. To speed up Gadget calculation, multiprocessing feature has been added and is utilized within the code. The code also now produces a text output of the results which can be zip compressed for distribution/sharing of results.

Utilizing the BC4 Coxeter group space, the adinkra_nxn_constructor.py code creates the 384 L sign permutation matrices. These L sign matrices serve as the building blocks of all Adinkras, given that any two of them satisfy conditions of a set Garden algebra equations. Once the 384 L matrices are created, for each L matrix the script builds a list of compatible matrices that satisfy Garden Algebra conditions. For 384 L matrices, counting all possible matrix position permutations, there is a grand total of 36,864 Adinkras with N=4 or four color, four open node and four close node.

For calculating the Fermionic Holoraumy matrices the fx_vij_holoraumy.py script is used. This script takes the input of 36,864 Adinkras and for each Adinkra generates a set of six V Holoraumy matrices. This script can generate the $\tilde{V}$ Fermionic Holoraumy matrices and the V Bosonic Holoraumy

matrices as well as calculate the elle coefficients for $\tilde{V}$ matrices. This depends on the calculation configuration/setting used in run_adinkra_calc.py script. Within fx_vij_holoraumy.py there are three functions responsible for handling these tasks. The fermionic_holomats function calculates a set of six $\tilde{V}$ matrices for a given Adinkra. The bosonic_holomats function calculates the set of six V matrices. The calc_vij_alphabeta function uses the six $\tilde{V}$ matrices and alpha beta matrices to calculate the corresponding $\tilde{\ell}$ coefficient values which are used later in the calculation of Adinkra x Adinkra Gadget values. Once the calculation is finished executing the fx_vij_holoraumy.py script returns an array (list in Python) of calculated Holoraumy matrices or $\tilde{\ell}$ coefficients.

The New Gadget value calculation is done by fx_mpgadgets.py script. This script utilizes the multiprocessing features available in Python and this is reflected as 'mp' in the script name. The task of calculating Adinkra Gadget values is cumbersome because each Gadget calculation takes two Adinkras (they can be the same) this means there is a total of 1.36 billion possible Gadget calculations. Therefore multiprocessing was used as a way to significantly speed up Adinkra calculations over the BC4 space. This is executed by splitting the calculation into n sets, with $1 \leq n \leq 64$.

The fx_mpgadgets.py can calculate the New Gadget or Gadget mk. 2 value in three different ways. First is taking the sum of all traces of V matrices from any two Adinkras. This is performed in newgadget_mtraces and newgadget_trace functions. These functions are not multithreaded and only return the Gadget II values to caller. The second way is using the $\tilde{\ell}$ coefficients in the calculation of Gadget II values in an almost similar fashion to calculation of Gadget values. There is no difference in Gadget II values calculated by the Trace method or the $\tilde{\ell}$ coefficient method. The advantage to using the $\tilde{\ell}$ coefficients is that it eliminates extensive matrix dot product calculations and instead simplifies each gadget calculation to a sum of multiplied -1, 1 and 0. This type of Gadget II calculation method is done in the mp_gadgetcalc_abonly function. This function utilizes numpy multiprocessing module to split the Gadget II calculations into multiple threads (using up to 64 threads) and can also write the Gadget II values for any Adinkra to a text file. Each Adinkra's Gadget II values are saved to a numbered text file which makes for 36,864 text files. Because of this the text output is designed to be split into eight seperate folders, each holding 4608 text files