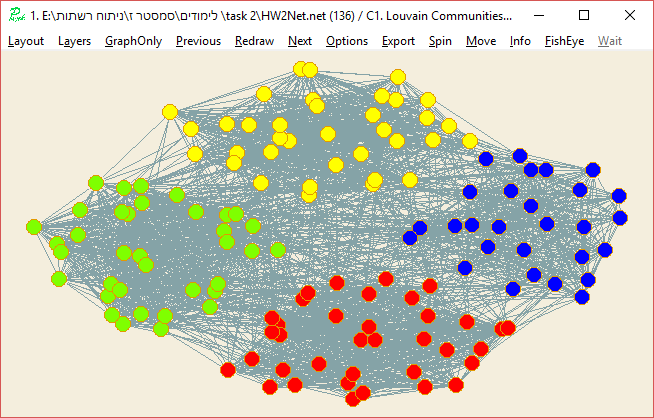
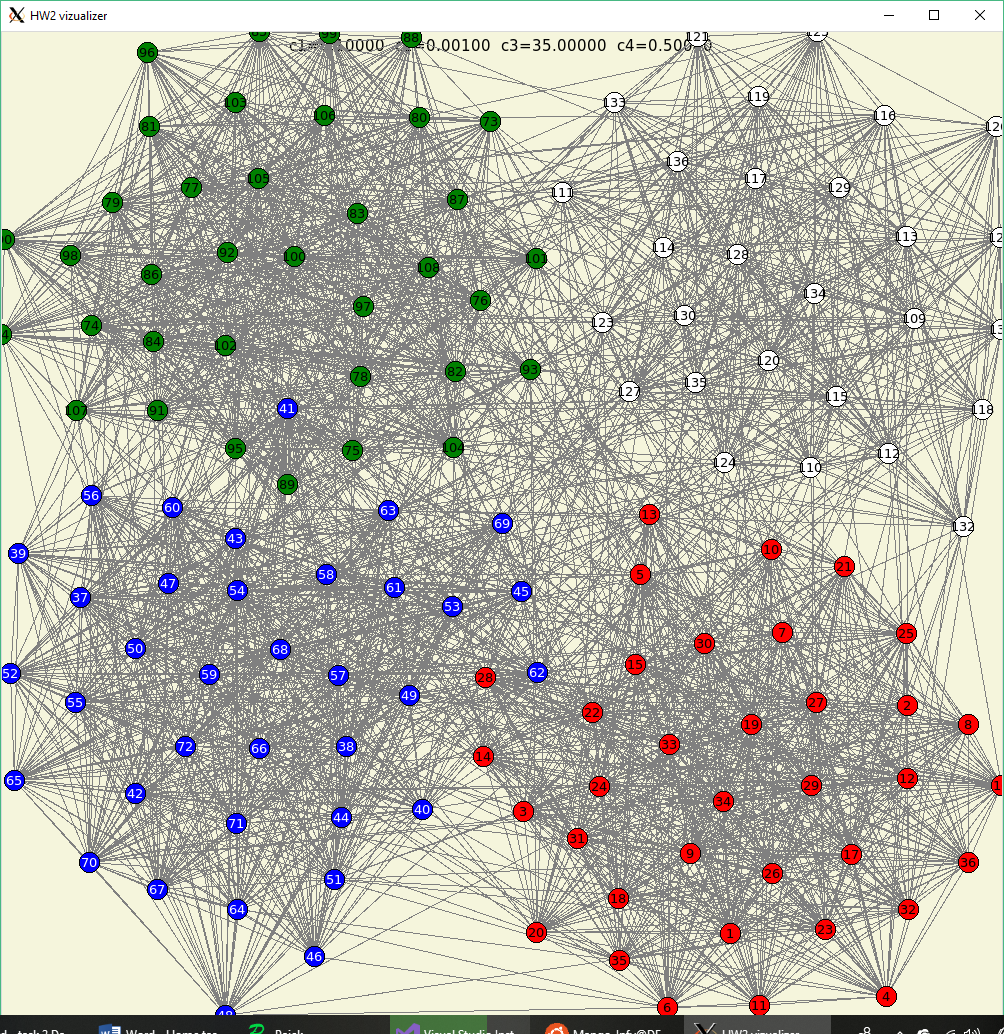
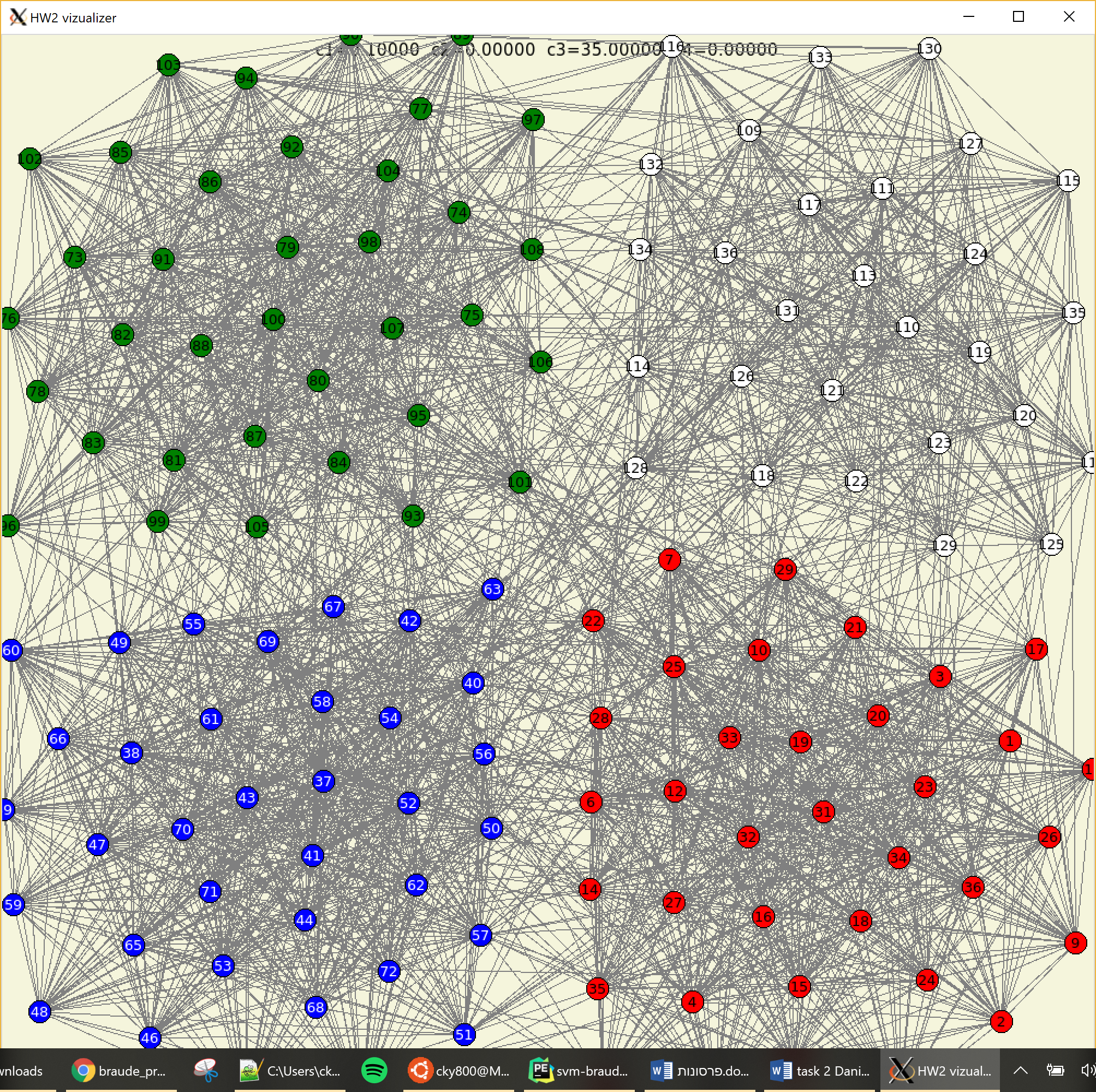
HOME TASK 2

1. My ID ends with 36 therefore I used 136 nodes and 4 clusters (6-2). See attached network created from MATLAB script.
2. Pajek output of net:



My final output nets using all criterions:

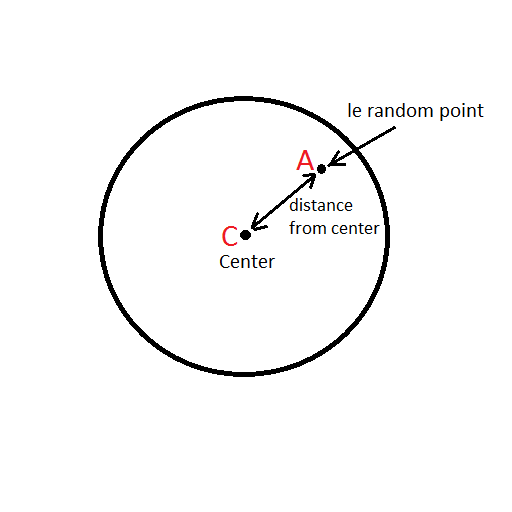
My output using only disregarding circular energy ( and edge intersection ( count.



1. Explanation of work:

Energy (cost) calculation using 4 different calculations:

*Explanation of :*

the way we calculate is so that the further it is from the center of the circle the more energy it produces.

Explanation of :

The way we calculated if 2 edges intersect is smart and simple. The idea was learned from the website <http://www.geeksforgeeks.com>

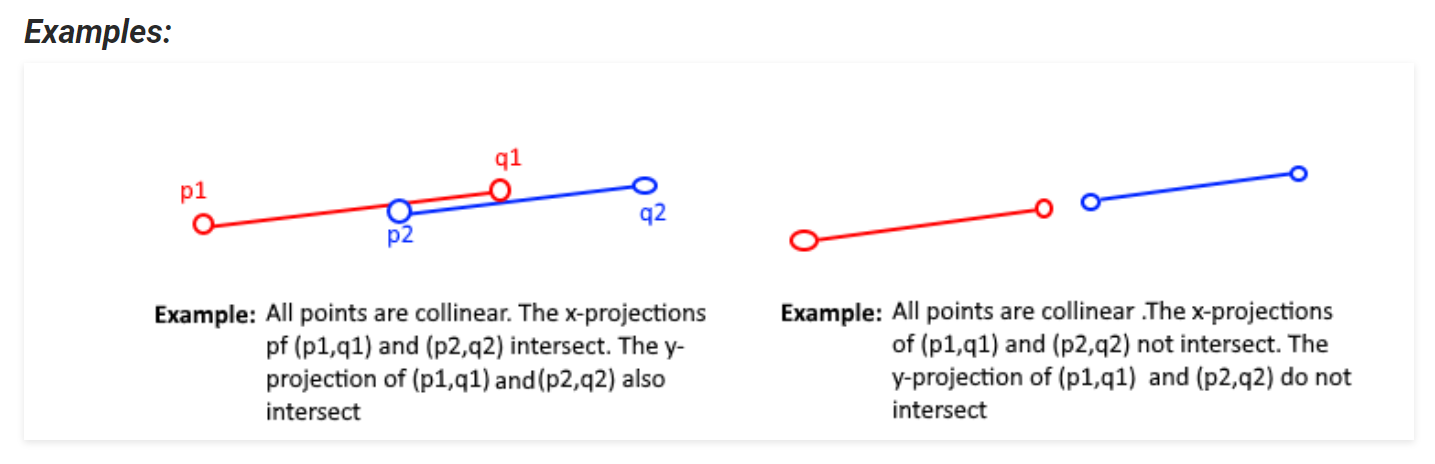
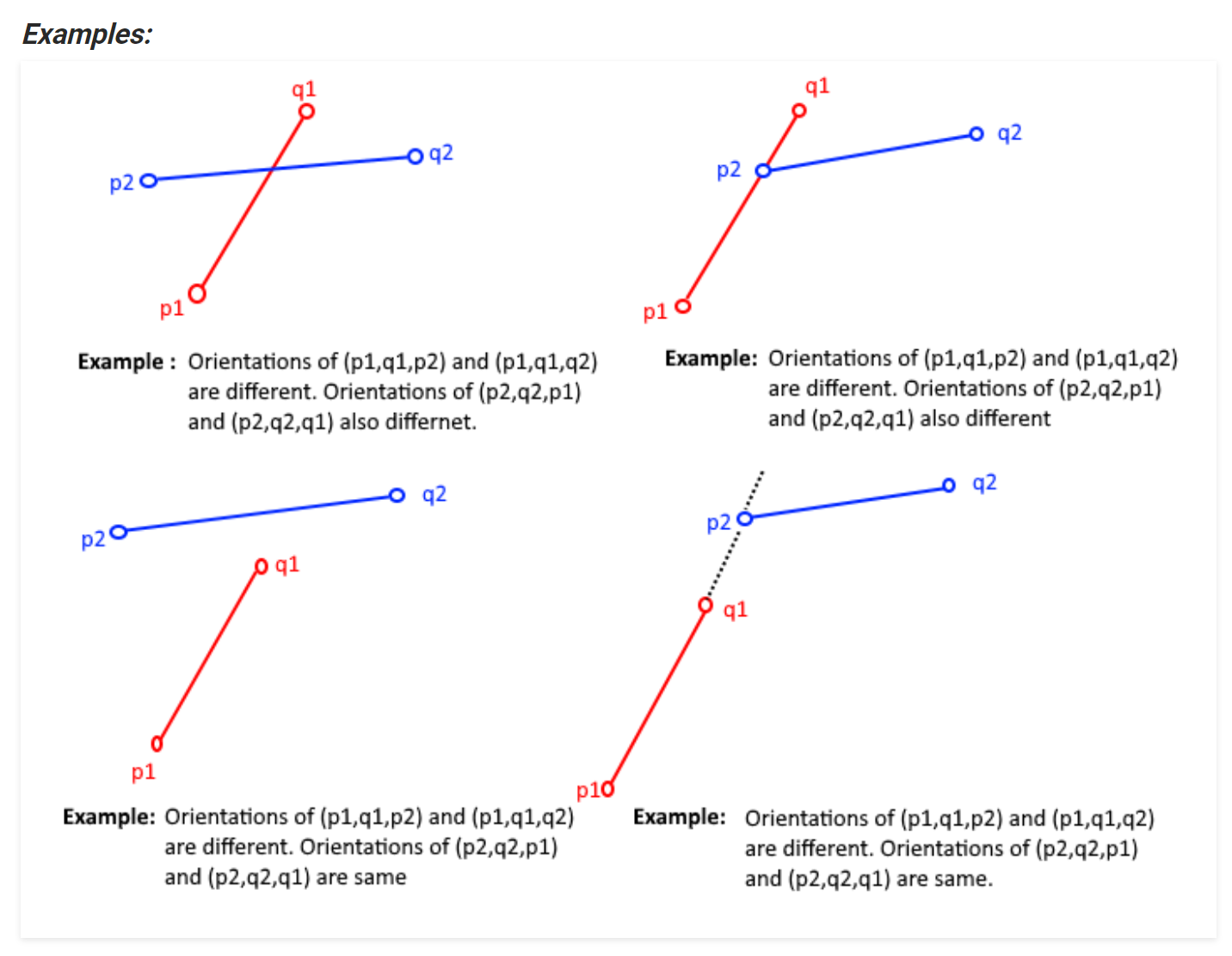
This is the only part of my code that was outsourced was the part where we check intersections.

It basically checks orientation of 2 segments and makes decisions if the segments intersect based on the orientation of the edges. (<https://www.geeksforgeeks.org/check-if-two-given-line-segments-intersect/>)

3 functions used: on\_segment, orientation, edges\_intersect.

The way it checks is by covering all cases:

Explanation of : simply sum the length of all edges.  
Explanation of :sum distance between all existing pairs



**Flow of work:**

**TCL IMPLEMENTATION:**

At first, I implemented the whole task in TCL (a scripting language I use at work a lot, so it was easier to implement in this language at first). TCL is a simple script language that considers everything as a string. It has hashmaps that are very useful and writing the energy function with it took about 2 days to implement + fix bugs.

After I realized what was going on I noticed that it is taking forever to finish the calculation and also that when it finished it did not work as I wish it did because of my Temperature and Constants did not have the correct values. I started with Temp=5000 and cooling rate of 0.95 for each iteration. 5000 was very high and did not work well and for this test it took all night to run

So I understoon I must implement in a more low level programming language.  
**C IMPLEMENTATION:**

I copied my tcl code to a new file and translated it to C. This was done on a Linux machine. My C code used TCL for its visualization. So I used fork() function to create a child process that will run the TCL code to show the graph results.

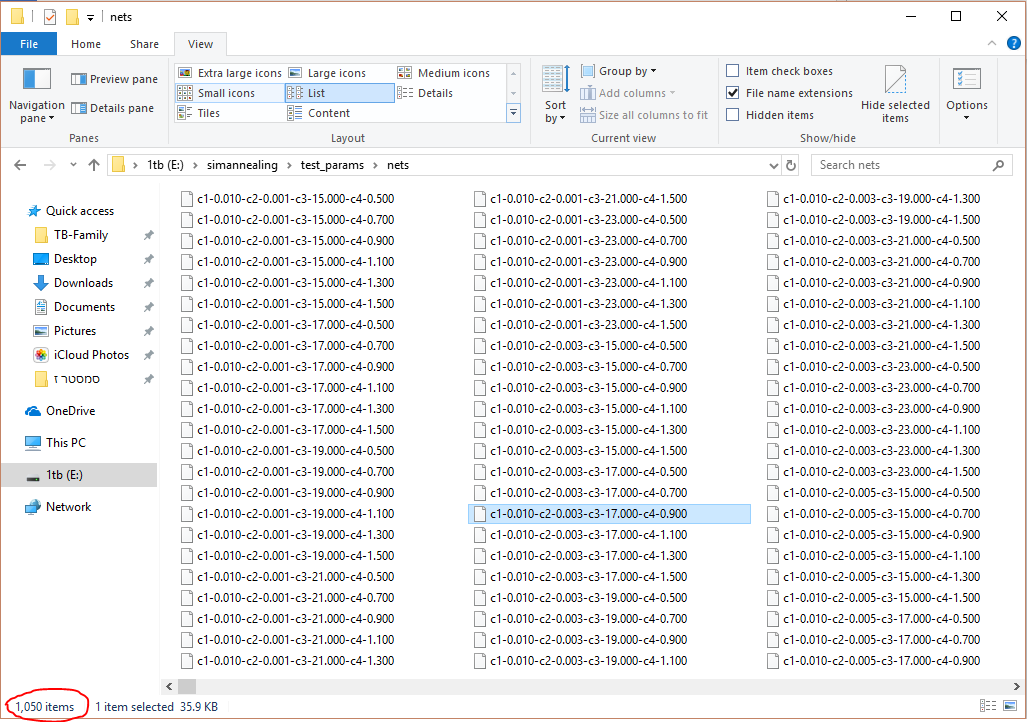
After investing another 2 days on rewriting the code in C and fixing bugs. The total runtime took only 45 seconds! But still the temperature and constant values were far off what they should have been.

**TEMPERATURE AND CONSTANTS:**

I asked around and found out that 5000 was a huge number and 100 does much better work  
I tried to figure our what should be the 4 constants for an optimal graph output, and it was not easy (this could not be discussed with others because this depends on the implementation).

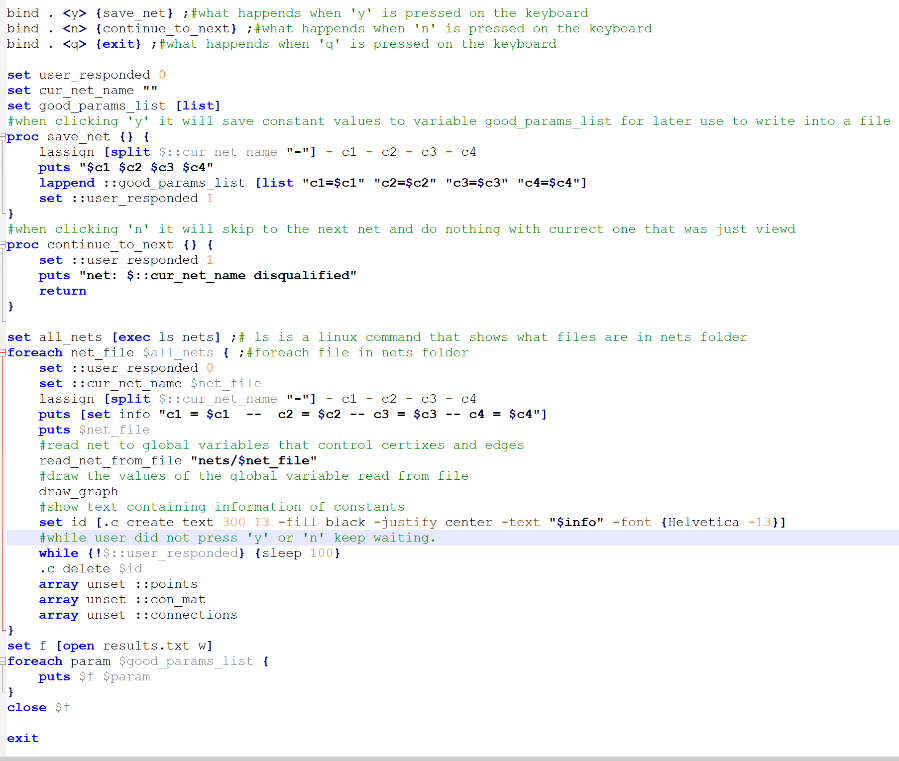
Because I wanted to save time on finding optimal constants for my energy function, I wrote a C program that uses 10 fork’s every 1.5 minutes each fork will calculate the results to a different value of constants: see following loop:



This Loop ran over night and generated 1050 nets that were results of different constant values 

After having all these nets the Automation process was not over!

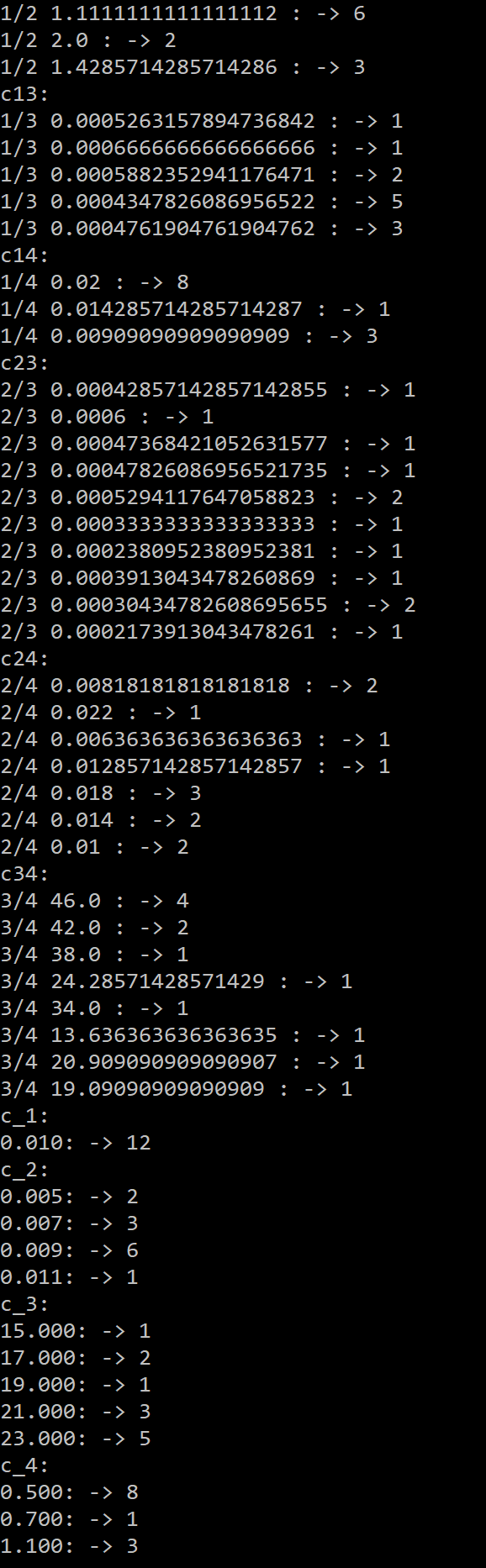
TCL CONSTANT FINDINGS:

I wrote another script that will show me each graph and by clicking ‘y’(for yes) or ‘n’(for no) I selectively choose all the constants that produced a fairly good graph. This means I used my brain and filtered all 1050 to good and bad results. See following implementation

When I was done, I analyzed the information by calculating the average constant values in order to choose those that shoed good results most of the time.

A script that did the calculations for me and spit out the bucketlist results:

An example of the output was:



Showing what values were most liked and what ratio between values was best.

This run is on part of the data and not all of it for the demonstration of this document.

Finally, I found the best values that fit my algorithm:

The most interesting part is the calculation of the total energy of the system and its complexity:

So in pseudo code would be:

\*\*\*note that we calculated edge intersection

The complexity of this energy function is

Note that all calculations were made using resolution 1.0x1.0 so all (x,y) values were between 0 and 1.

Only after creating a real resolution screen using TCL it was mapped to the size of the screen resolution.

FOR DETAILS ON CODE READ THE README.md file in the zip file