To run the visualization tool, do the following steps:

1- Install Eclipse enterprise edition, you can download it from (http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/junosr2).

2- Install Apache Tomcat, you can download it from (http://mirror.its.dal.ca/apache/tomcat/tomcat-7/v7.0.69/bin/).

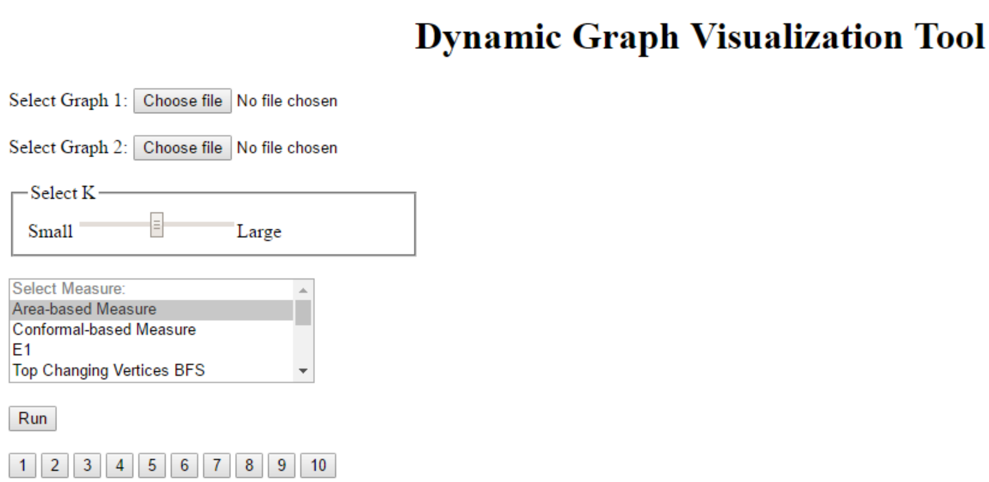
3- After downloading the Apache Tomcat, copy the jar "matlabcontrol-4.1.0.jar" found in lib folder at github repository to the tomcat lib folder.

4- Add MATLAB to the environment variables, as follows, go to start, right click on my computer, then properties. Choose Advanced system settings from the right, then choose Environment Variables. Choose PATH from the upper table, then press Edit, then add ";" and then add the path to your MATLAB installation bin folder. In my case the path is "C:\Program Files\MATLAB\R2014a\bin".

5- Open Eclipse EE and then create "new->other->Web->Dynamic Web Project" and name it "Dynamic Graph Visualization Juno" and then in the same window, press "New Runtime" choose "Apache Tomcat v7.0" , then next and choose the path you installed the tomcat to, so in my case it was "C:\apache-tomcat-7.0.69" and then copy the src from the github repository to the eclipse code and copy the WebContent from the github repository to the eclipse code and copy the lib folder too. Then right click on the "project -> properties -> Java Build Path -> Libraries -> Add JARs" and select from the lib folder "matlabcontrol-4.1.0.jar".

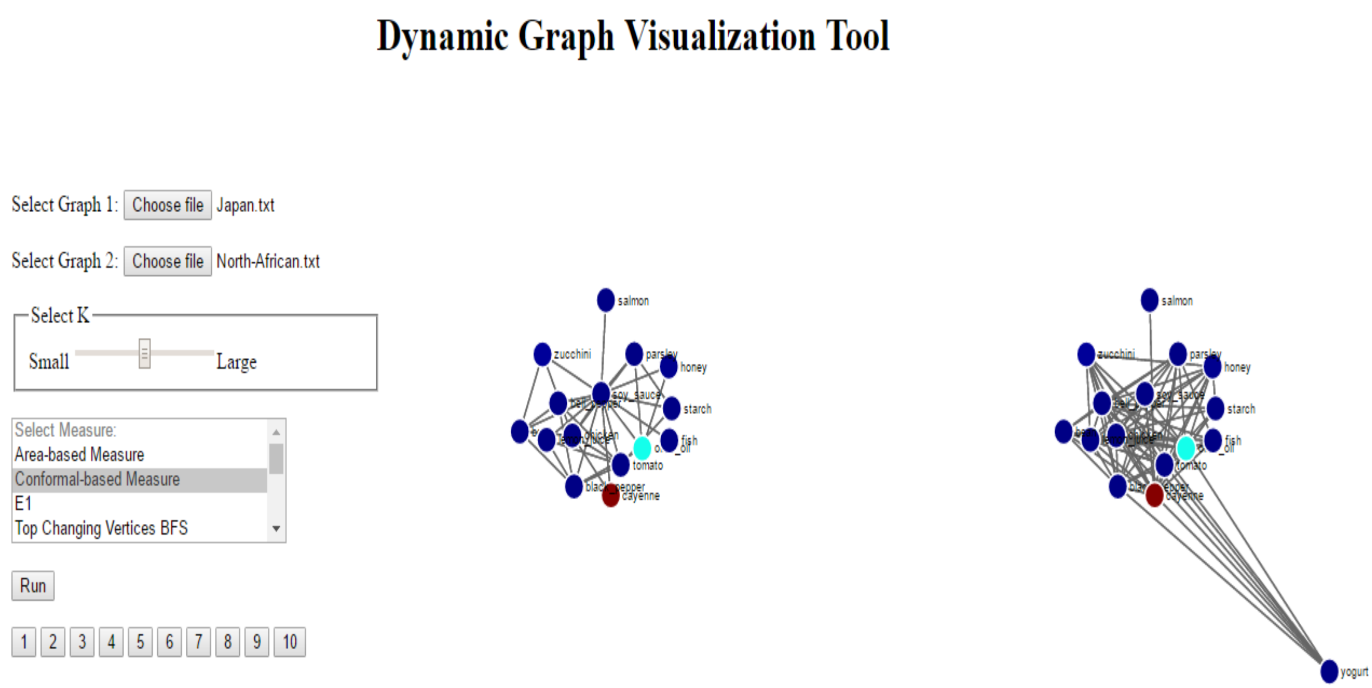
6- To run it, right click on the project -> Run As -> Run on Server, then open your Browser to the following URL:

"http://localhost:8080/Dynamic_Graph_Visualization_Juno/browser/dynamic_graph_page.html" and you will find the html page of the tool where you can select the files, change the k and the measure and run the code.



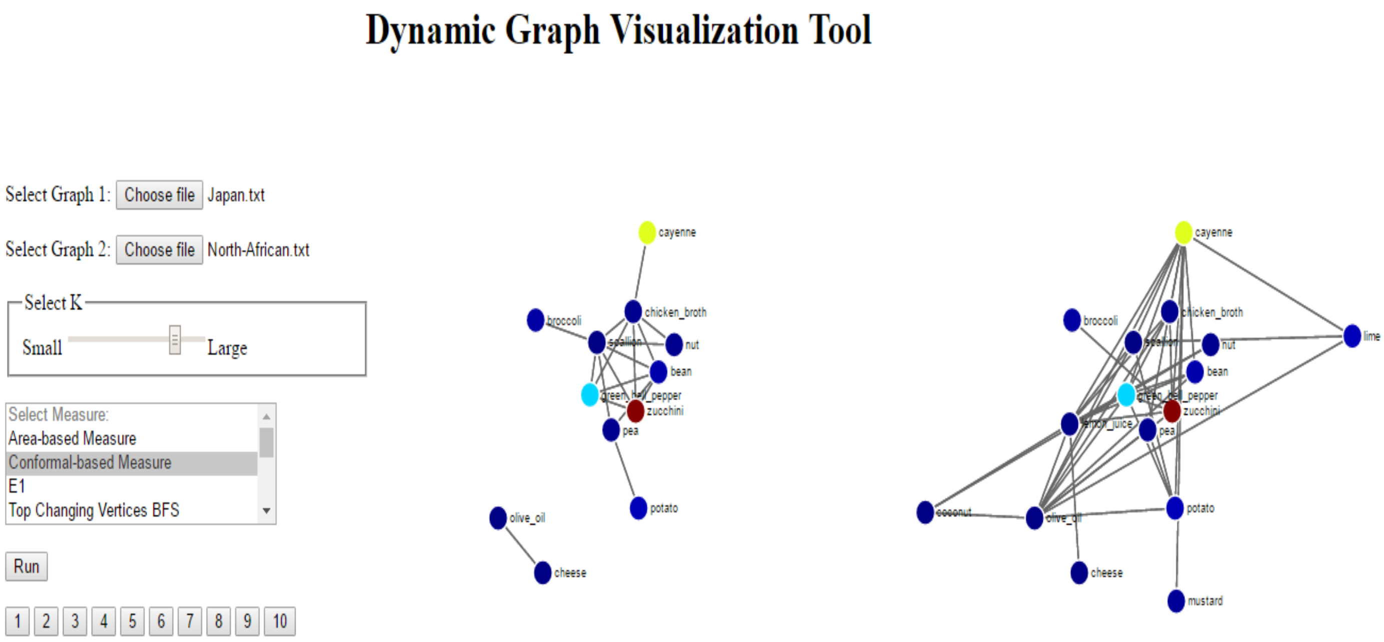
Then choose graph 1 file as "jars\data\Related Work Format\Food_graphs\Japan.txt" and graph 2 as "jars\data\Related Work Format\Food_graphs\North-African.txt" and then choose the suitable method and then press run and on the right, it will show you the highly distorted regions on both graph 1 and graph 2. Then, Matlab will open automatically and highly distorted region will be shown.

If we choose method to be conformal-based, the result will be:



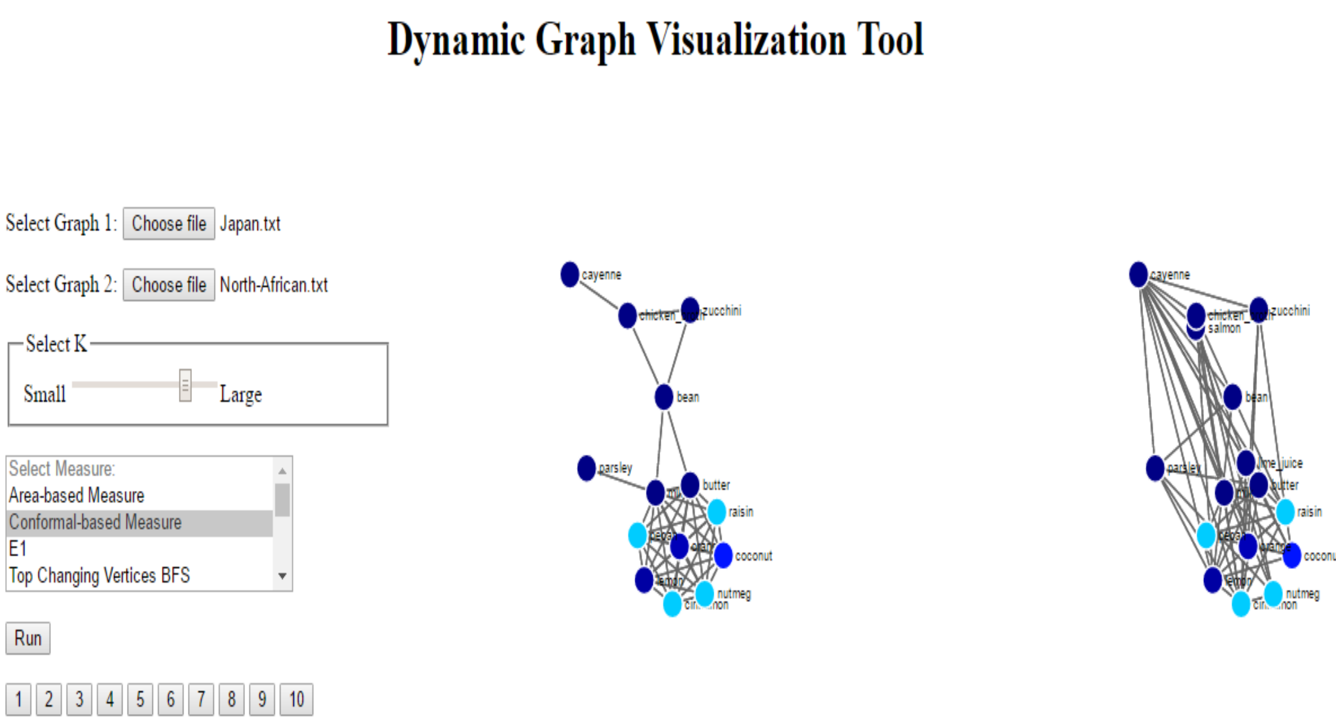
If you want to change the k value, use the scroll bar in the right to increase it or decrease. By default, changing k will change the colors on the selected region. To run from the scratch to get the highly distorted region using the k you have chosen, you have to press run or click on 1.

Here I have increased the k value:



To get the second highly distorted region press 2, with same analogy you can get the third distorted region by pressing 3 and so on for the top 10 highly distorted regions.

Clicking on button 2, we get:



The tool supports spectral clustering using area-based measure, conformal-based measure, E1 energy.

Also, it supports other related work methods, which are the following:

- Top Changing Vertices BFS
  For each vertex, calculate the following change measure:
  $$\text{change(v)=} \sum_u |G1(u,v) - G2(u,v)|$$
  Where G1 is the adjacency matrix of graph 1 and G2 is the adjacency matrix of graph 2. Then sort the vertices according to their change measure from the highest to the lowest and then pick the top r changing vertices and for each one of these top vertex performs BFS from this vertex until you traverse x vertices. Finally, each top vertex and its BFS reaching vertices will be the regions returned by this approach.

- Top Changing Vertices BFS Biased
  Same as the previous approach, but instead of doing traditional BFS from each vertex, we do a biased BFS as follows: Instead of adding all the neighbors of node n to the BFS queue, node n sorts its neighbors according to their change measure from the highest to the lowest and only add the top k unvisited neighbors to the BFS queue (k was set to 5 in all of our experiments).

- Top Changing Vertices BFS with Priority Queue
  Same as the first approach, but instead of using a traditional queue, we use a priority queue, where the priority of the node is equal to its change value. The higher this change value, the higher the priority of the node.
- Top Changing Regions BFS
  For each vertex, calculate the following change measure:
  $$\text{change(v)=} \sum_u |G1(u,v) - G2(u,v)|$$
  Where G1 is the adjacency matrix of graph 1 and G2 is the adjacency matrix of graph 2. Then do BFS from every node, store the returned region. After that, sort the regions according to their change measure (The used measure is $\left(\frac{\sum_{v \in Region} chang(v)}{\min(\max(\text{number of edges in R in graph 1,1}),\max(\text{ number of edges in R in graph 2,1}))}\right)$ from the highest to the lowest and then pick the top r changing regions that have size equal to the desired number of nodes per region.

- Top Changing Regions BFS Biased
  Same as the previous approach, but instead of doing traditional BFS from each vertex, we do a biased BFS as follows: Instead of adding all the neighbors of node n to the BFS queue, node n sorts its neighbors according to their change measure from the highest to the lowest and only add the top k unvisited neighbors to the BFS queue (k was set to 5 in all of our experiments).

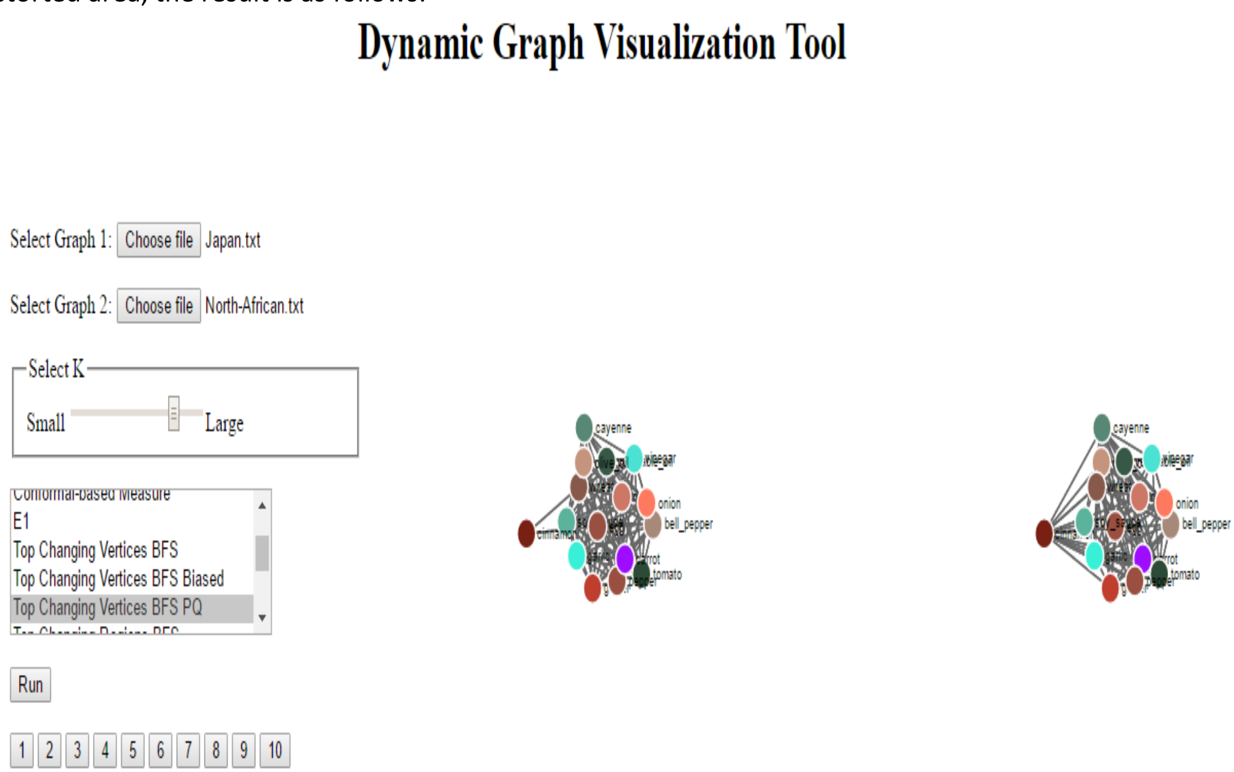
- Top Changing Regions BFS with Priority Queue

Same as the first approach, but instead of using a traditional queue, we use a priority queue, where the priority of the node is equal to its change value. The higher this change value, the higher the priority of the node.

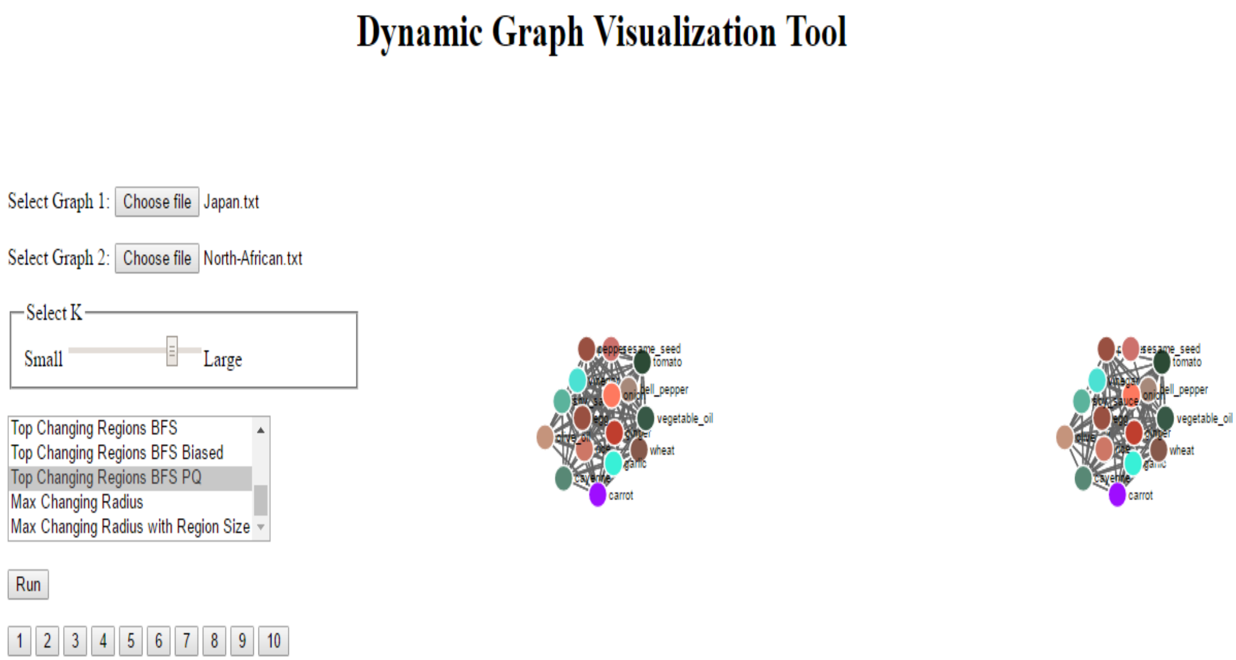
- Max Changing Radius
  Each vertex performs a BFS until reaching a radius a of the BFS graph. Then the approach sorts these regions according to the sum of the vertices change in them from the highest to the lowest and report the top r regions.

- Max Changing Radius with Region Size
  Each vertex performs a BFS until reaching a radius a of the BFS graph such that radius a is the smallest radius that return a region with at least number of vertices equal to x. Then the approach sorts these regions according to $\left(\frac{\sum_{v\in Region} chang(v)}{\min(\max(\text{number of edges in R in graph 1},1),\max(\text{ number of edges in R in graph 2},1))}\right)$ from the highest to the lowest and report the top r regions.

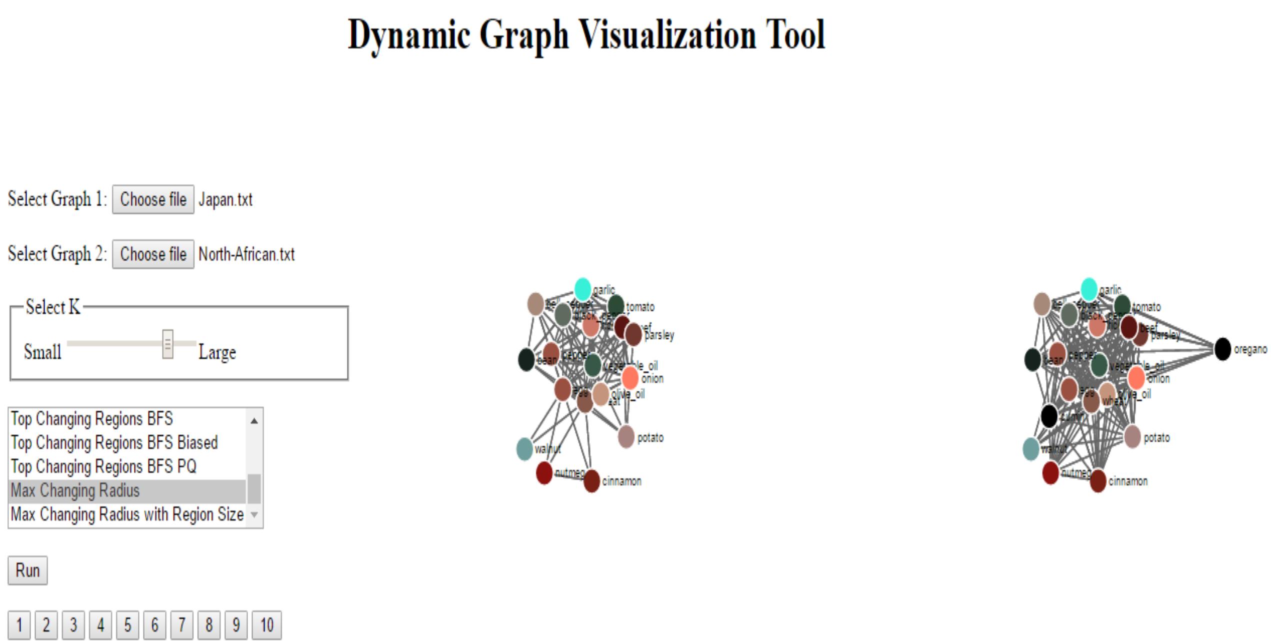
Lets run Top Changing Vertices BFS with Priority Queue on the food graph and get the highest distorted area, the result is as follows:



Lets run Top Changing Region BFS with Priority Queue on the food graph and get the highest distorted area, the result is as follows:



Lets run Max changing radius on the food graph and get the highest distorted area, the result is as follows:



If you have any question regarding the tool, please email me at
rania.ibrahim.salama@gmail.com or r7ibrahi@uwaterloo.ca