MCMC-of-Final-Year-Project-HKU

This repository mainly studies the Gibbs Sampler and vertex coloring problem (There is a connected graph which consists of $\mathbf{n}$ vertices, $\mathbf{e}$ edges and $\mathbf{q}$ colors are offered, we need to assign colors to each vertex such that no two adjacent vertices share the same color). Former scholars mainly focused on the minimum number of colors that could be used to color the graph, but here we focus on the application of Gibbs sampler on this problem. All the code was implemented from scratch.

What we are concerned about

About vertices

We call a certain valid way to assign colors to each vertex as one configuration. Assume all valid configurations are uniformly distributed, what is the average number of vertices which all occupy a same certain color (say, red)? Intuitively, we may think that the average number of vertices which occupy a same certain color should be the same among each color, which is $\frac{n}{q}$. Then how do we prove it? If there is a method to "traverse" the configurations, even if the traverse is not complete (i.e. do not traverse exactly every configuration), will the average number of vertices occupying the same color converge during the traversal?

About edges

We define the type of an edge by the colors of the two vertices it connects (e.g. an edge connecting two vertices of color 0 and color 1 respectively is called type (0,1)). If the assignment of colors is "randomized" enough (i.e. uniformly distributed), then the expected number of edges of type (x,y) should be $\frac{e}{qC2}$ where e is the total number of edges in the graph. How do we show this?

Reasons for using MCMC

Obviously, this is an NP-Hard question because when we increase the number of vertices or the number of colors, the number of valid configurations grows exponentially. Thus, it is really hard to count using combinatorics, which makes it really difficult if we wish to prove our assumptions above. MCMC gives a feasible way to tackle this issue. Assume $X_1 \dots X_N$ denote all valid configurations (note that N can be extremely large) and $h(x)$ is a function whose domain is $X_1 .. X_N$, then according to the theory that $E(h(x)) = \lim_{n \to +\infty} \frac{1}{n} \sum_{i=1}^{n} h(x_i)$, we can "simulate" the process by drawing a sample from the domain each time and continue this process for a large number of iterations and use the average value to estimate the expectation. However, the main problem is, how to do the sampling (i.e. get x_1 ... x_n)? According to the theory of MCMC, first of all, we need to construct a "transition mechanism" satisfying the Markov

property (i.e. $P(x_{i+1}|x_i, x_{i-1}\dots x_0) = P(x_{i+1}|x_i) \forall i \in N$). After that, we can run this Markov chain and get a series of samples.

Transition mechanism in Gibbs Sampler

Suppose $\mathbf{Q}$ is the set of all colors. Generally, when Gibbs sampler is applied to this problem, in every iteration, we randomly pick one vertex, search the set of colors of its neighbors so that we can fixed the set $\mathbf{Q}'$ of all the valid colors to this vertex. Then, we uniformly pick one color from $\mathbf{Q}'$ and update the color of this vertex. However, for simplicity, we can update the vertices sequentially rather than randomly. That is, in the $i-th$ iteration, we update the $(i \ mod \ n)$ th vertex. The updating mechanism remains the same.

The sufficient condition of MCMC is the $\mathbf{aperiodicity}$ and $\mathbf{irreducibility}$ of the Markov Chain. For aperiodicity, since $\mathbf{Q}'$ contains the current color of the vertex we are updating, the probability of staying in the same state is non-zero. Hence, aperiodicity is satisfied. For irreducibility, basically, the number of colors really matters. For example, if the graph is fully connected and $\mathbf{q} = \mathbf{n}$ (i.e. each vertex occupies a unique color and there is no more color. Recall that $\mathbf{q}$ is the number of colors and $\mathbf{n}$ is the number of vertices), then there is no way to update the configuration of this graph because any time you pick a vertex, the only valid color for this vertex is the current color and you cannot change it to a different one. Thus, here we set a secure lower bound for the number of colors. Denote $\mathbf{d}$ as the maximum degree among the vertices in the graph, let $\mathbf{q} > 2 * \mathbf{d}$. The following presents an algorithm to find a series of transitions from a configuration $\mathbf{A}$ to another configuration $\mathbf{B}$.

Suppose $\mathbf{V}$ is the set of vertices of the graph, then $\mathbf{A}$ = $\{a_{v_i}|v_i \in \mathbf{V}\}$ and $\mathbf{B}$ = $\{b_{v_i}|v_i \in \mathbf{V}\}$. Suppose $\mathbf{C}$ is the set of vertices which occupy different colors in $\mathbf{A}$ and $\mathbf{B}$ and let $|\mathbf{C}|$ = m (i.e. There are totally m vertices in the graph which occupy different colors in $\mathbf{A}$ and $\mathbf{B}$.) Let $\mathbf{A}'$ = $\{a_{c_i}|c_i \in \mathbf{C}\}$ and $\mathbf{B}'$ = $\{b_{c_i}|c_i \in \mathbf{C}\}$.

Algorithm:

$transitionAB \ (\mathbf{A}, \mathbf{B}, c_i)$:

$if \ c_i \ has \ been \ modified$:

$break$

$else \ if$ ($\mathbf{A} \setminus \{a_{c_i}\}$) $\bigcup \{b_{c_i}\} \ is \ valid$:

$\mathbf{A}$ = ($\mathbf{A} \setminus \{a_{c_i}\}$) $\bigcup \{b_{c_i}\}$

$else$:

$randomly \ pick \ one \ color \ which \ is \ not \ occupied \ by \ the \ neighbors \ of \ b_{c_i}$

$assign \ the \ picked \ color \ to \ a_{c_i}$

$transitionAB(\mathbf{A}, \mathbf{B}, c_j)$

$\mathbf{A}$ = ($\mathbf{A} \setminus \{a_{c_i}\}$) $\bigcup \{b_{c_i}\}$

Then we apply the algorithm to each vertex in $\mathbf{C}$ in order. Here are some notations to the algorithm:

1. The function is applied to one single vertex
2. In the third case, since we have assumed that $\mathbf{q} > 2 * \mathbf{d}$, we can always find a valid color which not occupied by the neighbors of b_{c_i}.
3. In the third case (changing the color of c_i to the objective color in $\mathbf{B}$ directly is invalid), then the adjacent vertex which occupy the objective color in $\mathbf{A}$ must be in $\mathbf{C}$. Or, $\mathbf{A}$ is no longer valid because $\mathbf{A}$ $\mathbf{B}$ share the same color assignment in vertices other than $\mathbf{C}$.

Using this algorithm, we can always find a series of transitions between any two valid configurations. Thus, the Markov Chain is $\mathbf{irreducible}$ and $\mathbf{aperiodic}$, which proves that the Gibbs sampler is a valid MCMC method to do the simulation task.

Objectives of doing this research

Now that we have proved that Gibbs sampler is a good method to do simulation for this problem, we wish to find something interesting related to vertices and edges. The process is mainly to initialize the coloring of the graph randomly (i.e. to generate X_0 randomly), then run the Markov Chain to generate samples X_1, X_2, X_3 ..... The state at time t (i.e. X_t) should be generated based on the posterior distribution given the state at time $t-1$ (i.e. X_{t-1}). Based on the samples, simulations can be done in a lower computational cost compared with traversing all valid configurations. Thus, as we have discussed before,

in terms of vertices, we will mainly discover:

1. For a certain graph, when changing the total number of colors we use or when changing the connectivity of the graph (i.e. to add some more edges to the original graph), if we only focus on the vertices occupying a same color, how will the speed of convergence of the average number of such vertices be affected?
2. If we focus on all colors and construct a $\mathbf{1} * \mathbf{q}$ vector, the i th entry represents the average number of vertices which occupy the color i. Then we focus on the average absolute distance between this vector and the vector of expectation (i.e. a $\mathbf{1} * \mathbf{q}$ vector in which every entry has value $\frac{1}{q}$). What will the speed of convergence of the average absolute distance be affected this time?

In terms of edges, we will mainly discover:

1. Will the number of each type of edges converge to $\frac{e}{qC2}$?
2. If so, will there be any difference in the speed of convergence between different colors?

Experiment

Construction of the graph

I implemented two algorithms to construct the graph:

Method 01

We first random shuffle the vertices list (i.e. [0, 1, 2, ..., n-1]). Then iterate the vertices list. Each time we pop the head of the list (return the first element of the list and delete it). For the ith vertex, randomly connect it with k vertices among the first i-1 vertices. k is a random integer between 1 and a threshold which is set as a hyperparameter.

Method 02

Since for every connected graph, we can always delete some edges so that it can become a tree. We can generate a tree (which is already connected) and then add some more edges randomly afterwards. When generating the tree, we also random shuffle the vertices list (i.e. [0, 1, 2, ..., n-1]). Then every time we pop the list and set the poped vertex as the right-most node of the current tree. The number of subtrees for each non-leaf node is randomly assigned. After the construction of the tree, we add some more edges to this tree.

Generally, Method 01 can generate a graph with larger connectivity (measured by average degree).

Initialization of colors

As we have stated before, denote the maximum degree among the vertices in a graph as $\mathbf{d}$, then for security, we let $\mathbf{q} > \mathbf{2} * \mathbf{d}$. Initially, we set Null to all vertices' colors. We assign the colors to vertices in a random order. (i.e. Random shuffle the list [0, 1, 2, ..., n-1]) Then we iterate the shuffled list. Each time we search the colors of the neighbours of the current vertex and uniformly choose one from all the valid colors. Since $\mathbf{q} > \mathbf{2} * \mathbf{d}$, there must exist available colors in every iteration.

Iteration

Basically, when running the Markov Chain, every time we need to select a vertex i randomly and update its color. Let $\mathbf{Q_i}$ be the set of colors occupied by the neighbors of vertex i. The new color is selcted uniformly from $\mathbf{Q}/\mathbf{Q_i}$. Thus, it is possible to stay in the same configuration after an iteration. Without the loss of generally, we update the colors of vertices sequentially (i.e. Update the color of vertex $i \bmod n$ in the i th iteration).

Methods to compare speed of convergence of two sequences

Since we are concerned about convergence, we need a practical method to compare the speed of convergence of sequences which are generated when running the Markov Chain. The method that I have tried are as follows.

Order of convergence

Mathematically, a practical method to calculate the order of convergence is to calculate the following sequence which converges to q :

$$q \approx \frac{log|\frac{x_{k+1}-x_k}{x_k-x_{k-1}}|}{log|\frac{x_k-x_{k-1}}{x_{k-1}-x_{k-2}}|}$$

For two convergent sequences, it is feasible to compare their speed of convergence by computing their order of convergence respectively as long as q converges for when k becomes large.

Ratio test

It is also feasible to employ ratio test when comparing the speed of convergence of two convergent sequences. Suppose there are two convergent positive sequences $\{a_n\}$ and $\{b_n\}$, if the sequence $\{\frac{a_n}{b_n}\}$ converges to 0, then $\{a_n\}$ converges faster. If it converges to any positive real number, then the speed of convergence for the two sequences are similar. If it diverges, then $\{b_n\}$ converges faster.

Show by plots

The most intuitive and straightforward way to compare the speed of convergence is by ploting the sequences and compare them directly.

According to the results of some tentative experiments, it is impractical to compute the order of convergence because there is always no such a q which is stable when k becomes large. Meanwhile, ratio test is also proved impractical because the sequence of ratio neither converge nor diverge to positive infinity. Thus, the only way we can use is to show by plots.

Initialization of the graph

Before doing the experiments, I tested the two methods for the initialization of the graph. The setting of hyper-parameters are as follows. Please note that this procedure aims to compare the two methods rather than finding the best hyper-parameters. Meanwhile, the setting of hyper-parameters is not the focus of this research:

Result of Method 01

Num of Vertices	Max connection with former vertices	Avg degree	Max degree
50	10	16.0	27
100	10	18.0	35
150	10	18.67	39
200	10	19.0	46

Result of Method 02

Num of Vertices	Max subtree	Max added edges	Avg degree	Max degree
50	4	8	2.28	5
100	4	8	2.14	6
150	4	8	2.093	6
200	4	8	2.07	6

Comparation of the two methods

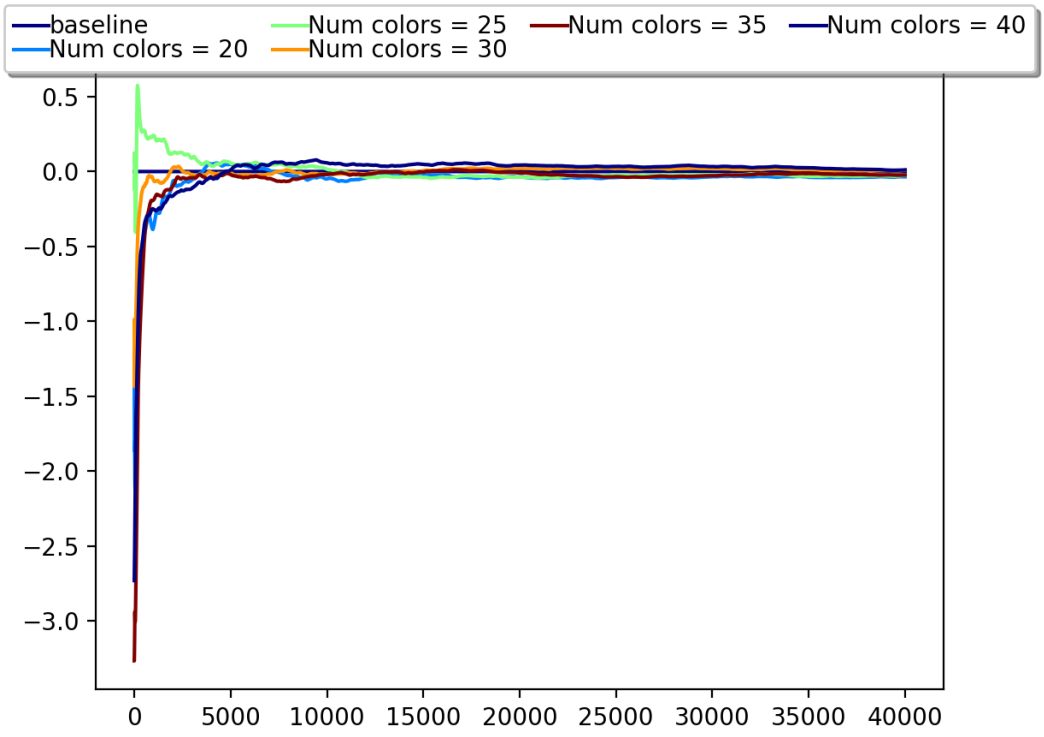
We can see from the statistics that method 01 initializes a graph with larger connectivity since the graph has larger maximum degree as well as larger average degree. Thus, we need to use more colors if method 01 is used to initialize the graph since our premise is $\mathbf{q} > \mathbf{2} * \mathbf{d}$. Moreover, method 02 is more "stable" because the average degree and maximum degree remains almost unchanged when we increase the number of vertices. Therefore, we employ method 02 for the following experiments.

Discovery 1: Focusing on one color

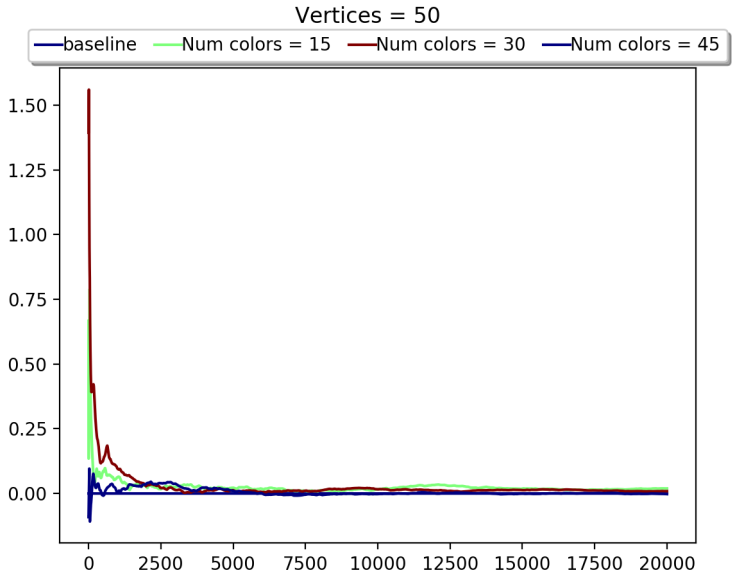
In this experiment, we wish to discover the average number of vertices which occupy the same certain color when running the Markov Chain. We denote n_{i_j} as the average number of vertices which occupy color i from the beginning to the j th iteration. The color index i is randomly picked from $\mathbf{Q}$ and it is trivial. Theoretically, the sequence $\{n_{i_j}\}$ converges to $\frac{n}{q}$. Since when we change q, $\frac{n}{q}$ will also change, it is necessary to adjust the sequences to $\{n_{i_j} - \frac{n}{q}\}$, which theoretically should converge to 0. Moreover, in order to reduce random error, for each setting of number of colors, we repeat the experiment for 50 times and use the average. As we have discussed before, we do the following two sets of experiments:

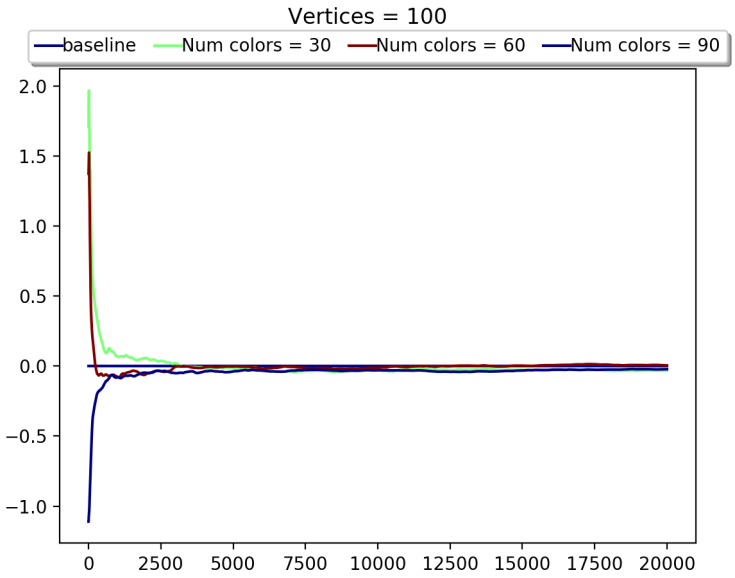
Change total number of colors

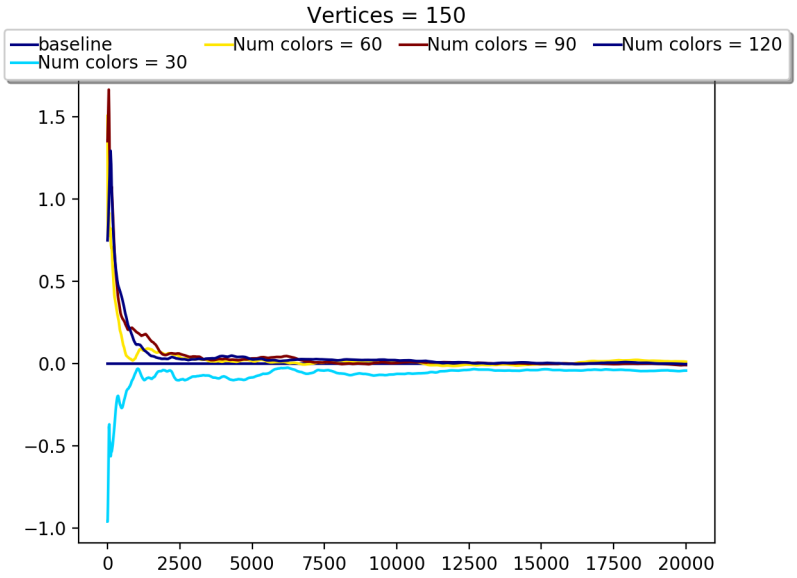
Basically, we try different number of colors on the same graph and the index of selected color is trivial. Firstly, I need to show that the Gibbs Sampler really produces a convergent result. I set the number of vertices as 150. The number of colors were 20, 25, 30, 35, 40. I get the following plot in which the x-axis is iterations and the y-axis is $n_{i_j} - \frac{n}{q}$.

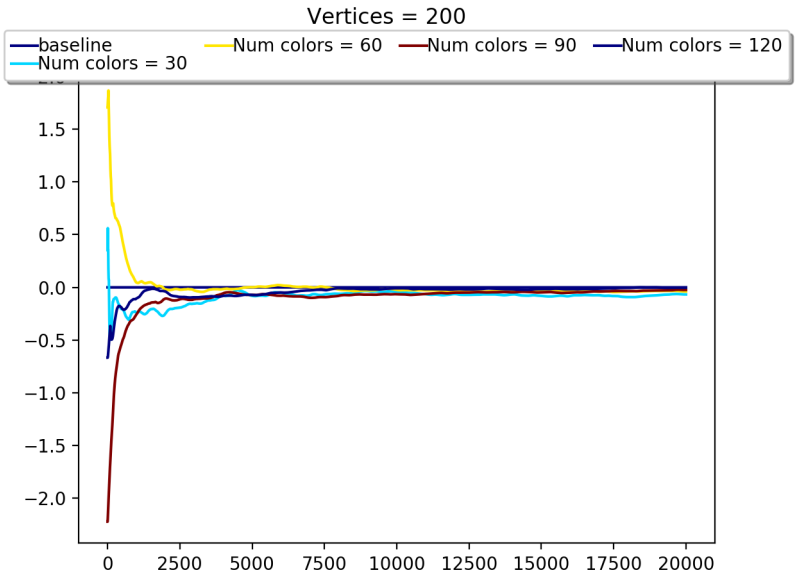


We can see that no matter how many colors we use, ultimately they will converge. However, the difference in speed of convergence is not obvious. We need to enlarge the difference in the number of colors. Meanwhile, the sequences have converged after 15,000 iterations, thus, we are able to reduce the number of iterations in order to shorten the running time. We did the experiments as follows:


Vertices = 50
baseline
Num colors = 15
Num colors = 30
Num colors = 45
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
2500
5000
7500
10000
12500
15000
17500
20000


Vertices = 100
baseline
Num colors = 30
Num colors = 60
Num colors = 90
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
0
2500
5000
7500
10000
12500
15000
17500
20000

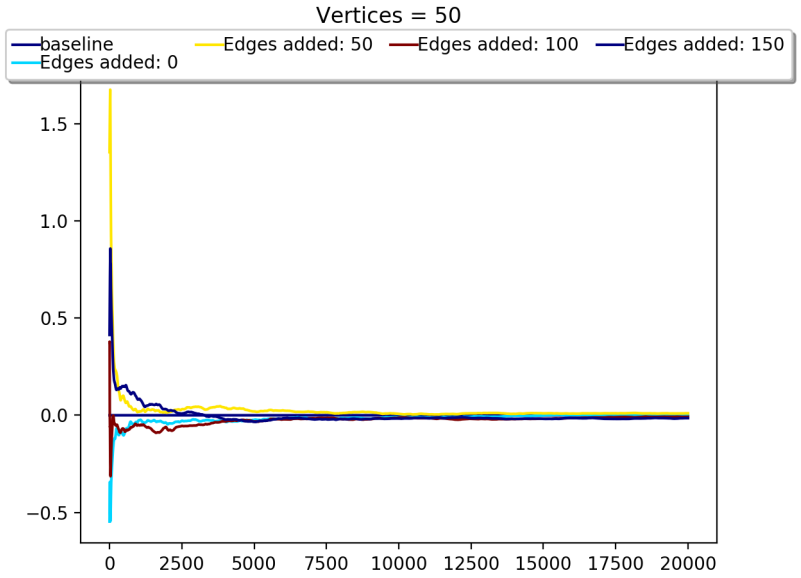


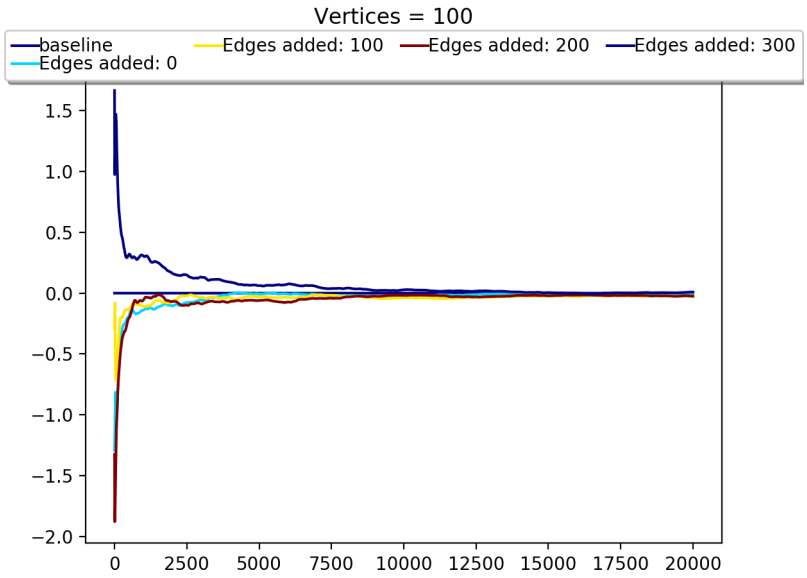


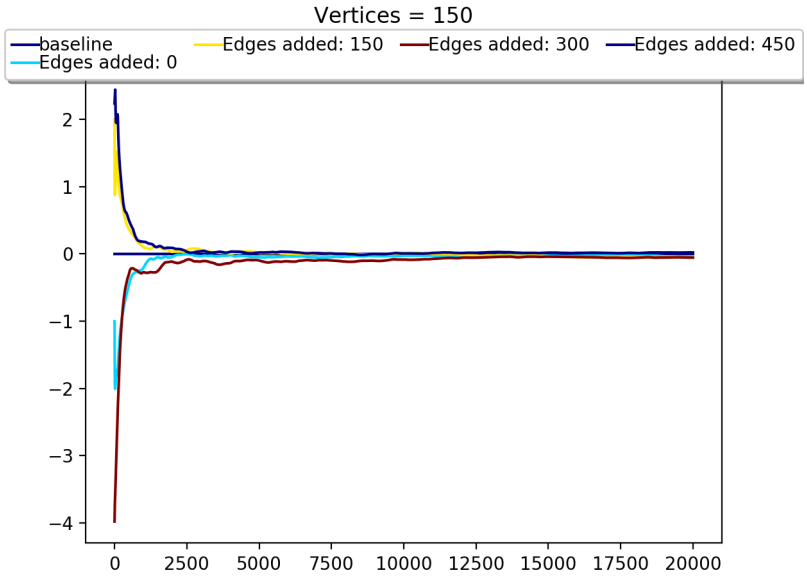
We can see from the plots that there exists a tendency of faster convergence when we increase the number of colors, it seems to converge faster. However, the evidence is not obvious because there exists some cases in which we increase the number of colors while the chain converges slower. Since we only consider the number of vertices occupying one certain color, random error is likely to affect the result.

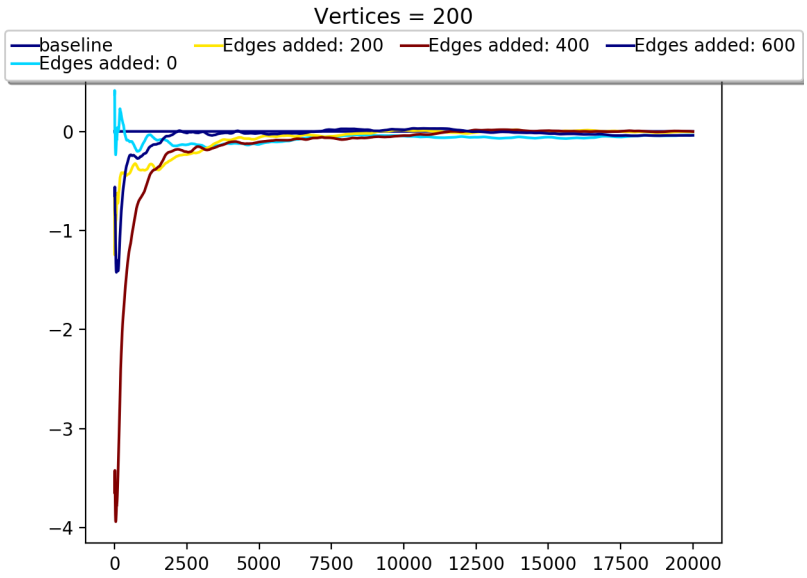
Change connectivity

Basically, we may add more edges to the existing graph so that the connectivity of the graph can be increased. If the conjecture in the previous section is true (i.e. more valid configurations, faster convergence), then when we increase the connectivity of the graph, there will be less valid configurations and the convergence should be slower. We conducted experiments on graphs with vertices 50, 100, 150 and 200. In each experiment, the number of edges that we add is proportional to the number of vertices. Here are the plots:


Vertices = 50
baseline
Edges added: 50
Edges added: 100
Edges added: 150
Edges added: 0
1.5
1.0
0.5
0.0
−0.5
0
2500
5000
7500
10000
12500
15000
17500
20000


Vertices = 100
baseline
Edges added: 100
Edges added: 200
Edges added: 300
Edges added: 0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
2500
5000
7500
10000
12500
15000
17500
20000





Roughly speaking, the light blue curve which represents the initial graph converges faster than the others. When we add more edges to the graph, they converge slower in different levels. Therefore, our conjecture is right. The more edges we add to the graph, the less valid configurations and the slower the convergence. Similar to the set of experiments above regarding the number of colors, the results are not stable. There are also some cases in which graphs with larger connectivity converge faster than those with smaller connectivity. Thus, we need to further reduce the random error. Since we have only focused on one color, we turn to focusing on all colors in the following experiments.

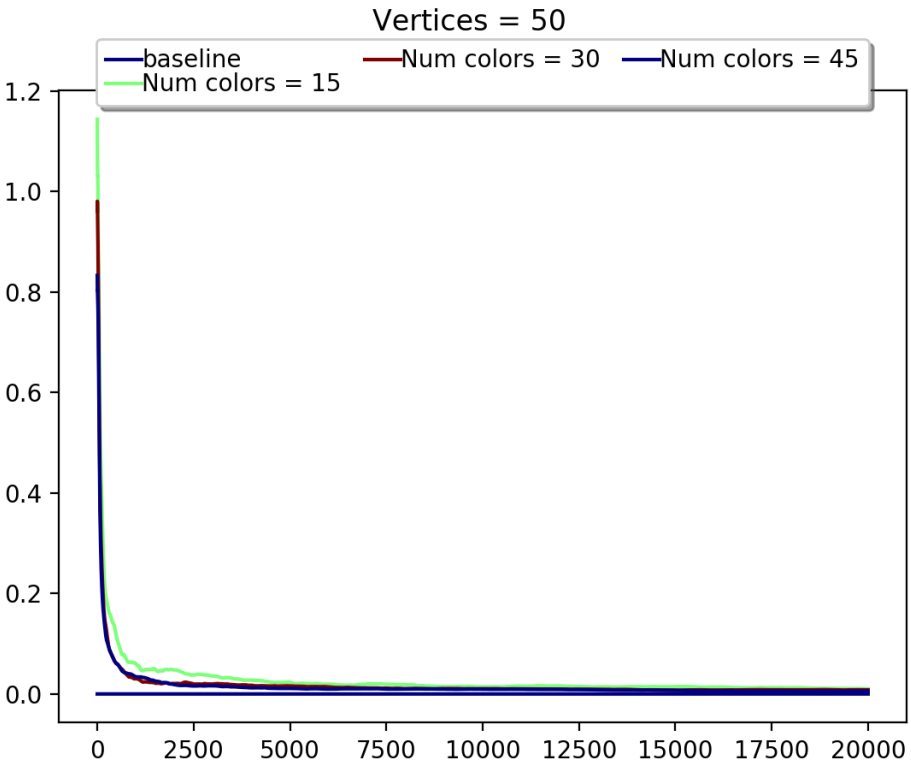
Discovery 2: Focusing on all colors

Now we focus on all colors. Theoretically, the expectation of the number of vertices which occupy any certain color equals to $\frac{n}{q}$. Thus, in this set of experiments, our idea is like this. We construct two vectors of length q, the first vector is called the expectation vector E in which each entry equals to $\frac{n}{q}$ and the second vector is called the reality vector R in which the i th entry is the average number of vertices which occupy the color i from the beginning to the current iteration. Thus, R is updated in every iteration when running the Markov Chain. We use R^j to define the R

vector in the j th iteration. We also define the "distance" between these two vectors as $d_{ER^j} = \frac{1}{q} * \sum_{i=0}^{q-1} |E_i - R_i^j|$. Theoretically, the sequence $\{d_{ER^j}\}$ should converge to zero. The advantage of using d_{ER^j} instead of the average number of vertices which occupy a certain color is that it eliminates the random error of one single color. We do the same thing as what we have done in Discovery 1.

Change total number of colors

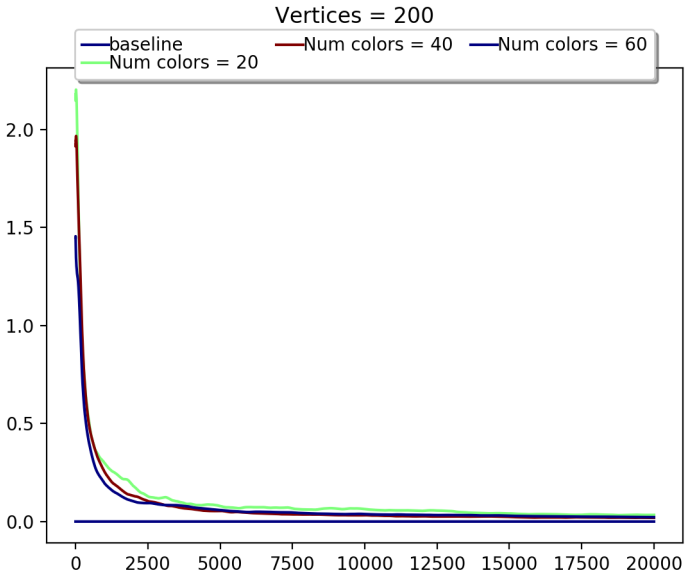
We set the number of vertices as 50 and the try 15, 30, 45 as the number of colors. We get the following plot:

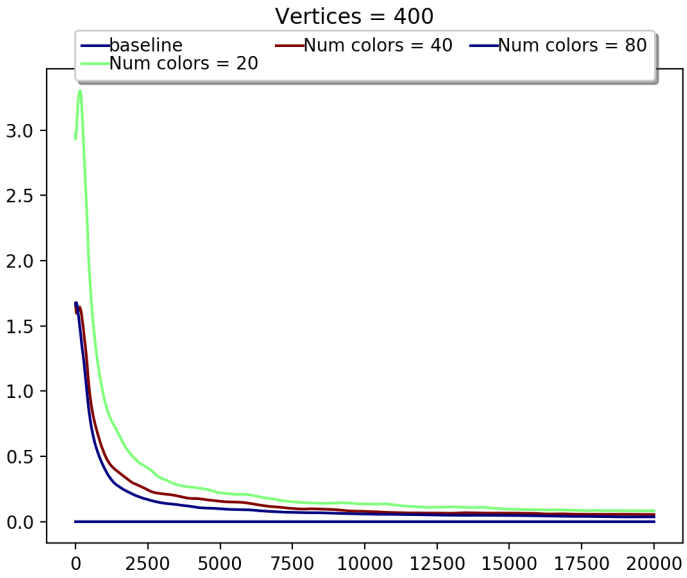


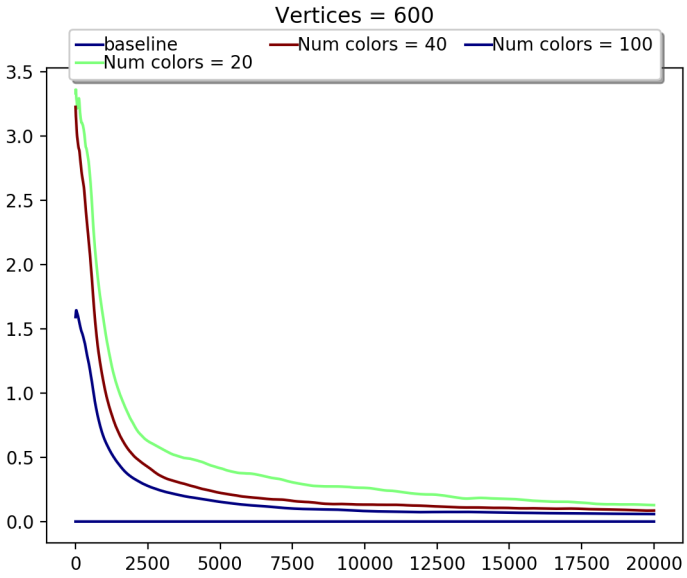
The plot illustrates that:

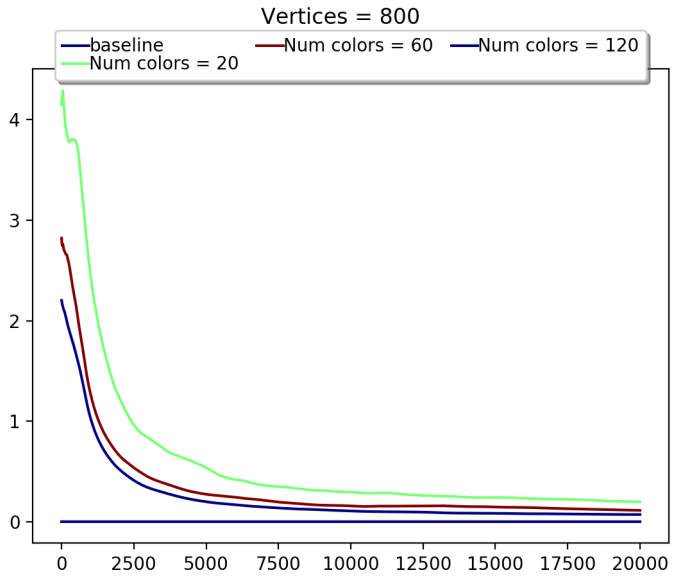
1. After using d_{ER^j} to measure, the sequences become much more stable because the oscillation becomes much smaller than what we got in Discovery 1
2. When we increase the number of colors, the sequence converges faster, which indicates the truth of our conjecture. However, the difference is not significant. We should increase the number of vertices so that we can enlarge the difference in number of colors.

Thus, we have to change the setting of vertices. Here are the final results:


Vertices = 200
baseline
Num colors = 20
Num colors = 40
Num colors = 60
2.0
1.5
1.0
0.5
0.0
0
2500
5000
7500
10000
12500
15000
17500
20000


Vertices = 400
baseline
Num colors = 20
Num colors = 40
Num colors = 80
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
2500
5000
7500
10000
12500
15000
17500
20000

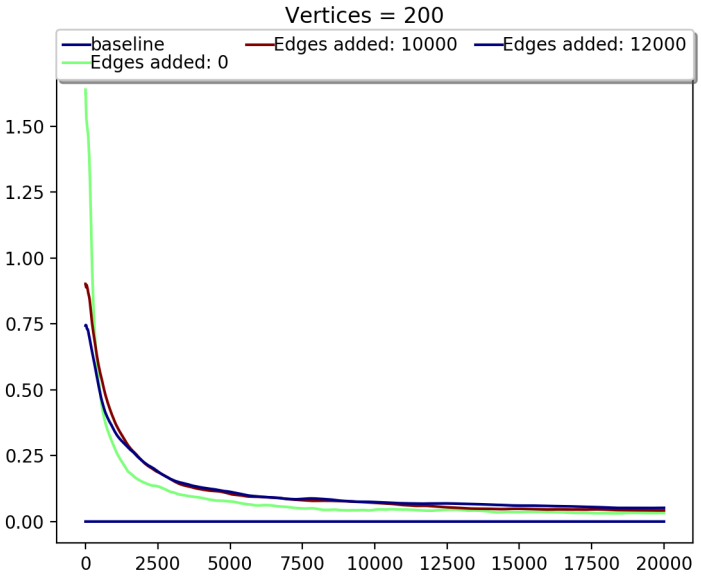


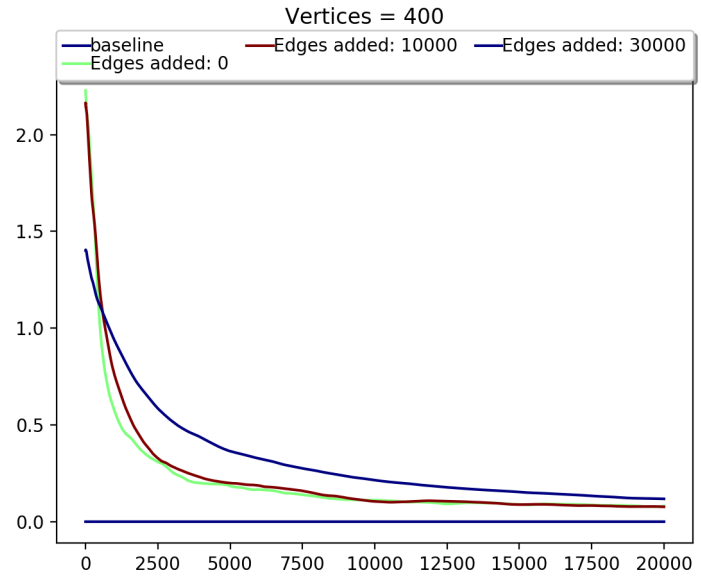


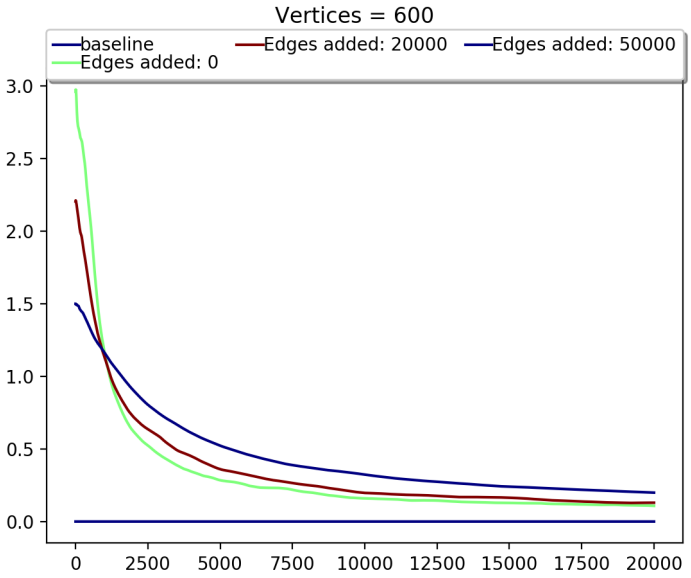
The four plots above show the truthfulness of our conjecture clearer. On the one hand, the sequences are more stable and they all converge to 0 smoothly. On the other hand, in every plot, the larger the number of colors, the faster the sequence converges. Apart from changing the number of colors, we also need to test the result of changing connectivity.

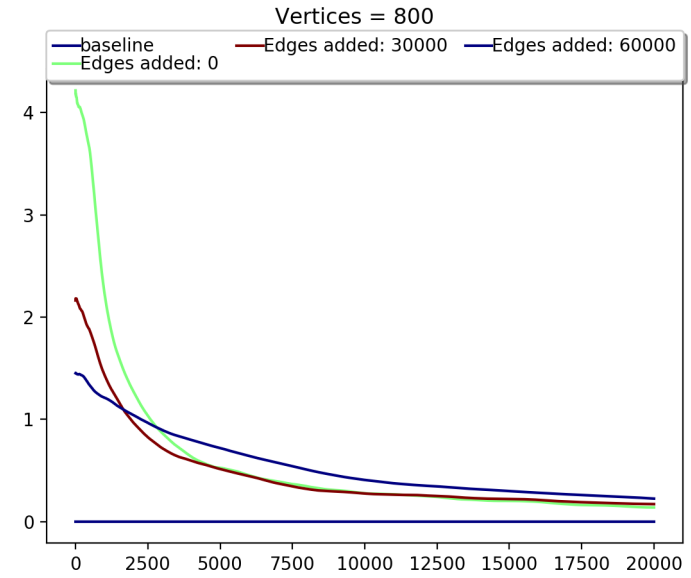
Change connectivity

Note that the average degree of the graph generated by Method 02 is around 2, so the total number of edges in the initial graph is approximately the number of vertices. And also, the connectivity of the initial graph is quite low. Thus, after some preliminary experiments, we add edges over 10000 in order to get sequences with significant differences. We change the connectivity and get the following results:


Vertices = 200
baseline
Edges added: 0
Edges added: 10000
Edges added: 12000
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
2500
5000
7500
10000
12500
15000
17500
20000


Vertices = 400
baseline
Edges added: 0
Edges added: 10000
Edges added: 30000
2.0
1.5
1.0
0.5
0.0
0
2500
5000
7500
10000
12500
15000
17500
20000





From the graphs above we can see that the sequences becomes much more smooth because of the less oscillation. In every graph, the light green curve representing the sequences for the original graph converges fastest. The dark blue curve representing the graph with the largest connectivity converges the slowest.

Conclusion of focusing on all colors

Based on our experiment results in both changing the number of colors and changing the connectivity, we find that:

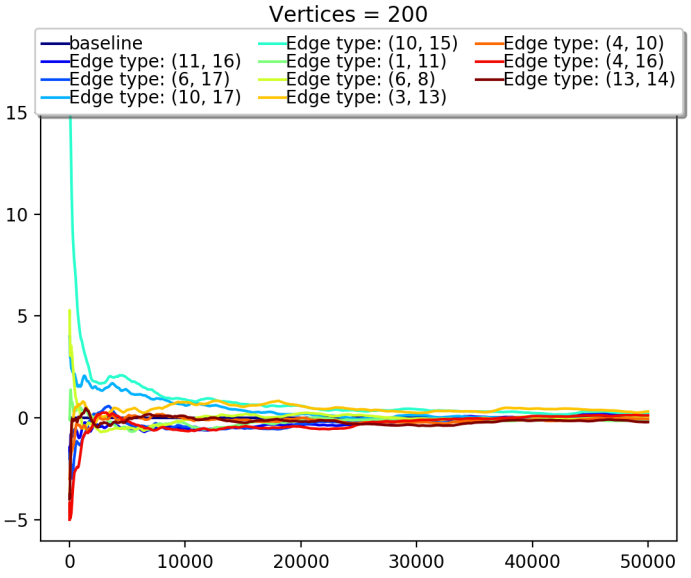
1. The more colors we use in the graph, the more valid configurations for the graph, the faster the chain converges.
2. The smaller the connectivity of the graph, the more valid configurations for the graph, the faster the chain converges

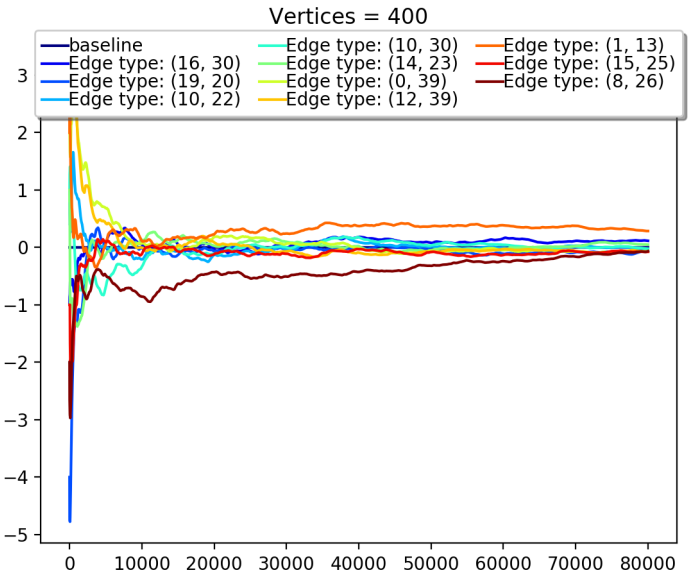
Therefore, we have the following generalized rule:

No matter what factor do we change, as long as it can increase the total number of valid configurations, the sequence of the average number of vertices which occupy the same color converges faster and vice versa.

Discovery 3: Focusing on edges

Now we change our focus to edges. This part dedicates to be a proof of the randomness of our former experiments. We define the type of an edge by the colors of the two vertices it connects. Then there should be $qC2$ types of edges. We propose a hypothesis that when running the Markov Chain for a large number of iterations, the average occurrences of each type of edge should be $\frac{e}{qC2}$ where e represents the total number of edges in the graph. Here we denote e_{i_j} as the average number of occurrences of edge type i from the first iteration to the j th iteration. For simplicity, we plot $\{e_{i_j} - \frac{e}{qC2}\}$. Then this sequence should converge to zero for any i. Moreover, since there are many types of edges, we just randomly choose 10 types of edges. Here are what we get:





From the plots, we can see that for any type of edge, when running the Markov Chain, the average occurrences of it converges to $\frac{e}{qC2}$. Thus, our hypothesis is true. This to some extents, proves the randomness of Gibbs sampler.

Conclusion

In this part, we are interested in applying Gibbs Sampler on the Vertex Coloring Problem. Basically, Gibbs Sampler is one kind of Markov Chain Monte Carlo algorithm which constructs an irreducible and aperiodic Markov Chain. The size of the transition probability matrix is always quite large and there is no need to know the exact size of the matrix. All we need to do is to have an initial state and transfer to the next state based on a certain probability distribution. After running the chain for a large number of iterations, we can use the average of a property of the samples generated by the Markov process to simulate the expectation of such property.

By the ideology of MCMC theories, our research objectives on vertices and edges have both been achieved. For vertices, we are interested in the average number of vertices which occupy the same color. After changing our focus from one certain color to all colors and take the arithmatic average of the absolute difference between the "hypothesized" expectation and the real average number of vertices in this color, we got smooth and convergent sequences. The generalized rule states that no matter what factor we change, if it increases the number of valid configurations, then the chain will converge faster and vice versa. For edges, results have shown that the average occurrences of each type of edge converge to $\frac{e}{qC2}$.