Graphs in ML project report : From degree distribution to graphs

Nicolas Rahmouni, Mhamed Hajaiej

1 Introduction

From a given graph, it's easy to extract the degree distribution which gives us some information about the properties of the graph. However, we sometimes need to do the opposite : generate graphs from a given degree distribution. This can be helpful for example when we are modeling complex real world network where we know the degree distribution. In that case, we would like to generate graphs and see if we can capture the properties of the real network or compute the expectancy of some observable over a whole class of graph. The number of possible graphs is potentially very large, even for a small number of nodes. We are interested in directed graphs and there is a few questions that we try to answer. What are the methods that we can use to generate the graphs ? What about their efficiency ? Is it possible to generate significantly different graphs from the same degree distribution ? And does the number of different graphs depends on the type of the graph (Erdos-Renyi, small-world, preferential attachment graphs)? In our report, we first present the different methods that we have tried for graph generation and their advantages and disadvantages. In the second part we deal with the question of graph similarity in order to evaluate the difference between the graphs that we generate. The code we used is available on GitHub at this link.

2 Generating directed graphs from a degree distribution

Our goal is to construct a simple graph (no self-loops and no multiple edges) given a sequence of in- and out-degrees. The most known method is an adaptation of the original Havel-Hakimi algorithm which only generates graphs in a subset of all the possible graphs with this degree sequence. We can then modify it to obtain other possible graphs but it's not clear if we can obtain significantly different graphs. Another method consists in using the configuration model, however this method is not efficient if we don't use approximations. Finally, we take a look at an algorithm that can construct all possible graphs directly, but at a high computational cost.

2.1 Havel-Hakimi algorithm

The Havel-Hakimi algorithm is an algorithm that permits to generate a single realization of an *undirected* graph which has a prescribed degree sequence. It is based on the following result which has been proven independently by V. Havel and S.L. Hakimi:

Theorem 1 (Havel [Hav55] and Hakimi [Hak62]) *There exists a simple graph with degree sequence* $d_1 > 0, d_2 \leq ... \leq d_n > 0$ *if and only if there exists one with degree sequence* $d_2 - 1 \leq ... \leq d_{d_1+1} \leq d_{d_1+2} \leq ... \leq d_n$.

This gives a simple greedy way to construct the graph when the sequence permits it. We just need to pick the node v_n and order the other nodes in decreasing order of degrees $\{v_{s(1)}, ..., v_{s(N-1)}\}$ where s is the result of an $argsort$ function. Then we construct the edges $\{(v_n, v_{s(k)}) \mid k \in [1, d_n]\}$ and finally, we update the degree sequence as prescribed by the theorem: $d_{s(k)} \leftarrow d_{s(k)} - 1$, $\forall\, k \in [1, d_n]$ and $d_n \leftarrow 0$. The algorithm stops when all degrees are zeros (in that case we have constructed a simple graph with the prescribed degree sequence) or if it is not possible to add all the edges because

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

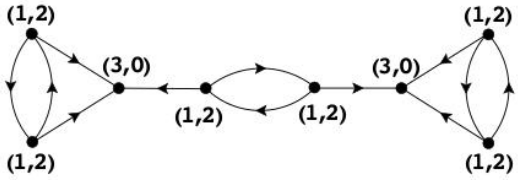


Figure 1: An example of a digraph that cannot be generated with the Havel-Hakimi procedure (extracted from [Kim+12])

the number of remaining non-zero degrees is lower than the degree of the considered node (in that case, the theorem states that it is not possible to construct a simple with the initial degree sequence).

2.1.1 Havel-Hakimi for directed graphs

Now, our problem is to generate a directed graph given a in and out degree sequence, or *bi-degree sequence* (BDS for short), (d^+, d^-) (d_i^+ is the out-degree of vertex v_i and d_i^- is the in-degree of vertex v_i). If such a graph exists, we say that the BDS is bi-graphical. For this problem, the Theorem 1 can be adapted to apply the exact same algorithm. But first let's define the normal order on a BDS which is the lexical order on all the vertex of the BDS, except the last one, where the first letter is d^-, the in degree, and the second is d^+, the out degree. Now the adaptation of the Havel-Hakimi theorem is the following:

Theorem 2 ([EMT10]) *Assume that the BDS (d^+, d^-) is in normal order, and that $d_n^+ > 0$ (n is the last vertex, which is not ordered). Then (d^+, d^-) is bi-graphical if and only if the residual BDS*

$$\Delta_k^+ = \begin{cases} d_k^+ & \text{if } k \neq n \\ 0 & \text{if } k = n \end{cases}$$

$$\Delta_k^- = \begin{cases} d_k^- - 1 & \text{if } k \leq n \\ d_k^- & \text{if } k > n \end{cases}$$

is bi-graphical.

Once again this gives a greedy algorithm to find a graph that realizes this BDS. We just choose a node v_n which has a non-zero out-degree, order the BDS in normal order, and connect v_n to the d_n^+ first nodes in the normal order. Then, we update the sequence of degrees as described in the theorem (it is the residual BDS after adding the edges) and loop over the vertices until all out-degrees are zeros or the residual BDS is not bi-graphical which happens when the number of nodes with a positive in-degree is lower than the out-degree of the considered node.

The complexity of the algorithm can be evaluated simply: at each step we have d_n operations (constructing the edges) and we sort the N vertices for a global cost of $\mathcal{O}(N \log(N))$ and we do N steps so the global complexity is $\mathcal{O}(N^2 \log(N))$ in worst cases. This can be a problem if we want to generate larges graphs ($N > 10^5$). The complexity can be reduced a little if we just update the order for the new BDS by taking into account that the initial BDS was sorted instead of brutally sorting the residuals.

Clearly, this algorithm is useful to generate one graph sample from a degree sequence, however some simple graphs realizing the BDS are not reachable with this algorithm (see for example Figure 1). In the next parts of this section, we describe other methods based on the same idea that permit to generate other kind of graphs than the ones generated by the Havel-Hakimi algorithm.

2.1.2 Modifying the graph to generate new ones

We can use edge swaps on the graph obtained by the previous method, in a Markov Chain Monte-Carlo way, to generate other graphs with the same distribution. The edge swaps can be done randomly. We can also do them in a preferential way that creates some structure based on a local criterion. For instance, we have tried an example where we disconnect some nodes from the rest of the graph.

In fact, any graph realizing a BDS can be obtained from another graph realizing the same BDS with at most $2|E|$ edges swaps where $|E|$ is the cardinal of the edge set (see Theorem 7 in [EMT10]). However, it's not clear how many swaps we need to obtain a significantly different graphs or if we can hope to obtain independent samples with this method. We study more this more in the second part of the report.

2.1.3 Preferential structure as initialization

This is another method that we have tried to obtain different graphs with Havel-Hakimi. Before applying the algorithm, we connected some edges and updated accordingly the degree distribution that we give to the algorithm as input. We can chose the edges randomly or in a way that it creates a certain structure. For instance, we have implemented a variant that creates a long-chain structure in the graph, and another one creating a long loop.
However, we have no guaranties that the obtained graph is graphical and we need to check for self-loops and multiple edges so the final result must be sampled with a rejection scheme (we accept the sample only if it is graphical, on the other cases we try sample another graph until it is accepted so the computation time can be very high).

2.2 Configuration model

If we have an in and out degree sequence (d^+, d^-), this method consists on assigning to every node v_i, d_i^- in-stubs and d_i^+ out-stubs. Afterwards, for every in-stub we chose uniformly at random an out-stub that has not been connected yet to connect to. Generating different graphs with this method gives independent samples. However, the obtained graph is not guaranteed to be graphical. We need to check it when the algorithm terminates. If it's graphical, we have our graph, otherwise, we have two solutions : either we reject the graph and we sample a new one, or we erase the self-loops and multiple edges to make it graphical. The rejection method may require a large number of rejections before finding an acceptable graph which is not efficient (a lot of time is wasted), especially for large graphs. The erase method is more efficient but it gives an approximation of the wanted degree distribution and not the exact one.
Giving some regularities conditions on the degree sequence, we have asymptotic guaranties (when the number of nodes $n \to \infty$) on the probability that the obtained graph is graphical.

There is also a method that generates a degree sequence that gives a graphical graph with an asymptotic probability of 1 if the distribution that we sample from satisfies some conditions. Finally, there are guaranties that the rejection and the erase method converge to the good distribution. We didn't implement the algorithm for generating the graphical degree sequence but more information about the guaranties are presented in[Olv12].

2.3 Independent graph sampling without rejection

This method is presented in [Kim+12] and provides a solution for sampling directed graphs with no self-loops nor multiple edge with an exact degree distribution. This method doesn't use rejections which makes it much more efficient than the Configuration model while keeping the advantage of generating independent samples. It also provides sample weights that are proportional to their sampling probability, which is helpful for statistical analysis.

The idea is to compute the exact set of all possible connection that will keep the sequence graphical for each considered node instead of simply taking a subset of this set like in the Havel-Hakimi algorithm. Indead, Thoerem 2 gives a set of nodes that will always give a graphical residual BDS but it is possible that another set of nodes also gives a graphical residual BDS. In principle, the algorithm works as the Havel-Hakimi algorithm: we take a work node with non-zero out degree, find the set of nodes to which we can connect the work node without breaking the graphicality of the BDS (this is the difficult part). Then, instead of adding all edges at once, we sample an edge from the possible set, compute the residual BDS and repeat those steps. The mathematical details are too long for this report but they are available in the article, here we are only presenting the algorithm

and we refer to some theorems of the paper.

Input Degree sequence $\overline{D}$ in normal order
Output Graph without self-loops or multiple edges with the exact distribution
1) Define as work-node the lowest-index node i with non-zero (residual) out-degree
2) Let χ_i be the set of forbidden nodes for the work-node, which includes i, nodes with zero in-degrees and nodes to which connections were made from i, previously. In the beginning, χ_i includes only the work-node and zero in-degree nodes.
3) Find the set of nodes, A_i that can be connected to the work-node without breaking graphicality. (This is the complicated step and it's detailed later)
4)Choose a node m $\in A_i$ uniformly at random and connect an out-stub of i to an in-stub of m.
5) After this connection add node m to χ_i.
6) If node i still has out-stubs, bring the residual sequence in normal order, then repeat the procedure from 3) until all out-stubs of the work node are connected away into edges.
7) If there are other nodes left with out-stubs, reorder the residual degree sequence in normal order, and repeat from 1).

Algorithm 1: Independent graph sampling without rejection pseudocode

Step 3) demands a finer analysis. Finding the acceptable set is done using the following assertions. First, let's note that for a work node v_i, if we define $\mathscr{L}_i$ as the set of the d_i^+ left-most nodes in the graphical order of the sequence $D = (d^+, d^-)$, and $\mathscr{A}_i$ the set of acceptable nodes for the work node, we have that $\mathscr{L}_i \subseteq \mathscr{A}_i$ by Theorem 2. Then, some results presented in [Kim+12] permit to find the other nodes in $\mathscr{A}_i$. In particular, they state that if a connection to a node breaks the graphicality of the sequence, any connection with a node at the right on the BDS in normal order will also break graphicality. This gives a good way to find the acceptable set: we just need to test the graphicality for all the nodes on the right of the set $\mathscr{L}_i$ in graphical order until graphicality is broken.

3.1) Reduce the out-degree at the work-node i and the in-degree at ℓ by unity, that is $\overline{d}_i^+ \to \overline{d}_i^+$ -1 and $\overline{d}_\ell^- \to \overline{d}_\ell^-$ -1 resulting in a new residual bds $\overline{D}_\ell$.
3.2) Bring $\overline{D}_\ell$ into normal order (required by Theorem 2 of[Kim+12]). 3.3) Add ℓ' to the forbidden set for the work-node.
3.4) Now, as required by Theorem 2, reduce by unity the in-degrees of the nodes in the left-most adjacency set $\mathcal{L}_{i'}$, and reduce the out-degree of the work-node i' to zero. This results in the new sequence $\overline{D'}_{\ell'}$
3.5) Order the bds $\overline{D'}_{\ell'}$ by in-degrees, non-increasingly.
3.6) Apply the FR theorem to test for graphicality (see [Kim+12] for the FR theorem).

Algorithm 2: Details of step 3

Clearly, the samples obtained using this algorithm are independent (each sample corresponds to a "path" of connection sequence which is chosen independently at random every time we run the algorithm), but this sampling is not uniform. Typically, this can be an issue when one wants to compute the expectancy of some function over the class of graphs generated by a degree sequence. However, this issue can be solved by normalizing the observation by the probability of sampling the graph. This probability of following a certain path in the accepted sets which can be computed easily during the execution of the algorithm (see [Kim+12] Section 4).

However, the complexity is in $\mathscr{O}(N^3)$ which becomes impracticable on a regular machine when the number of nodes is too high (typically 1000).

2.4 Summary on the sampling methods

So far, we have seen that there are three main methods to sample a graph with a prescribed degree sequence.

The first idea is to sample one graph with the Havel-Hakimi algorithm and then swapping the edges to obtain any other realization. The procedure has the advantage of being simple and pretty fast (the Havel-Hakimi has a complexity of $\mathscr{O}(N^2 \log(N))$ in the worst cases and the swaps are done in $\mathscr{O}(|E|)$). However, with a random sequence of swaps, the sampling is not uniform and more importantly the independence of the model is not well controlled. Furthermore, if we want the graph to have particular properties (long chain structure, long loops, etc.) we can construct at first this structure and apply the Havel-Hakimi algorithm on the residual degree sequence.

Number of nodes	50	100	200
Havel-Hakimi + swaps	10.2	41.1	309
Direct graph sampling	10.6	70.8	538

Table 1: Average computation time to generate 100 graphs with a given number of vertices for two methods.

If we want to directly sample a graph from all the possible ones, we can use the configuration model which has the advantage to sample uniformly. However, it uses a rejection scheme which can take very long as we reject most of the proposed graphs because of the self-loops and multiple edges.

On the other hand, we can sample a graph directly from the set of simple graphs with a prescribed degree sequence using the method of [Kim+12]. The advantage is that the sampling weights can be computed and the procedure does not involve rejection. However, the time complexity is higher than the Havel-Hakimi algorithm ($\mathcal{O}(N^3)$).

For a comparison of the computation time for the simple Havel-Hakimi algorithm and for the more elaborated procedure that directly samples the graphs, see Table 1. We can observe that as expected, the computation time for the direct sampling grows much faster with the number of nodes.

3 Graphs similarity measures and generation of different graph from the same degree distribution

3.1 Graphs similarity measures

The question of graph similarity can still be considered as an open question as it didn't receive any definitive answer. A lot of metrics can be considered and their proprieties and efficiency may vary from a graph to another[BW12], but most of the techniques includes the computation of some statistics over the graph or using the adjacency matrix. Some of these metrics can be very general and not sufficient to capture all of the graph particularities like the average of neighbors degree. In our analysis we have tried a few of them. In this paragraph, the metrics that we are using are two metrics based on the adjacency matrix : the hamming distance and the average of the values of the eigenvector associated to the highest eigenvalue of the matrix[WF94], another metric is the degree correlation distance[New02] suggested to have a high variation and potential for classifying different types of graphs by[BW12], and two metrics operating on histograms of the graphs Dijkstra distances and page rank.

3.2 Comparison of the variance of the graphs generated with different methods

In this section we focus mainly on comparing the variance of the graphs generated with the edge swaps and the one obtained with the independent sampling algorithm. First, we try to estimate how many swaps we need to obtain a significantly different graph from the original method using the edge swaps method. We can compare the pairwise distances matrix between the original graphs and the swapped version for various number of swaps, and we can observe that the metric saturate after a certain number of swaps. In Figure 3 we show the Hamming distance (averaged on 100 samples) for a number of swaps rising from 0 to 400. We also compare the original graph and the swapped version with other graphs generated with the same DD in order to check that the metric values after a high number of swaps are equivalent to the metric measure between two graphs generated independently from the same DD. In Figures 4, we show the result of the metrics obtained pairwise between two graphs, with 10 version of each graph obtained by edge swaps (the maximum number of edge swap is 100 here). This plot is obtained for graphs with 50 nodes and is averaged on 50 samples. The images can be seen as a 2×2 grid where the top left square and the bottom right square shows distances between graphs obtained from the same graph obtained by Havel-Hakimi on a DD but with a different number of swaps. The other 2 squares compare the distances between graphs obtained from the 2 different DD.Naturally, The diagonal represents a null distance. We first notice the difference obtained by the different distances used, which shows the difficulty of measuring graph similarities. We can see However that the distance gets higher as we get far from the diagonal (when we have made more swaps) in general. And that for 100 hundred swap (middle of the first row) we are almost at the same distance from the original graph as we are from

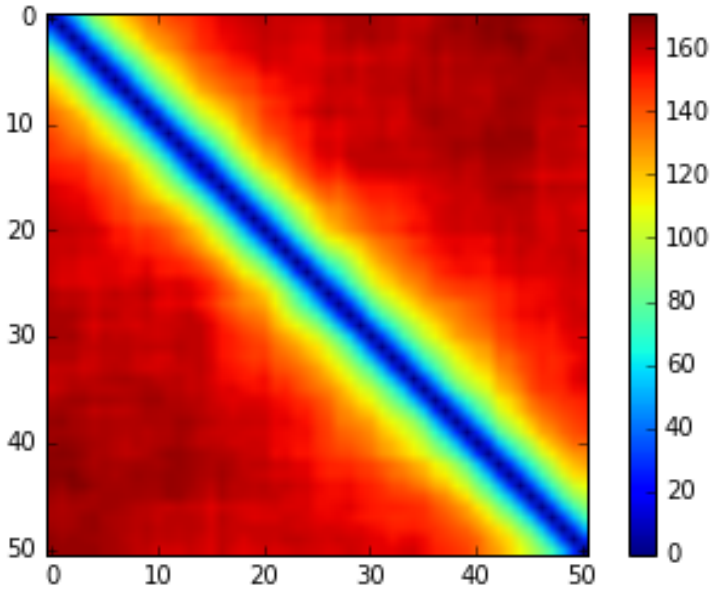
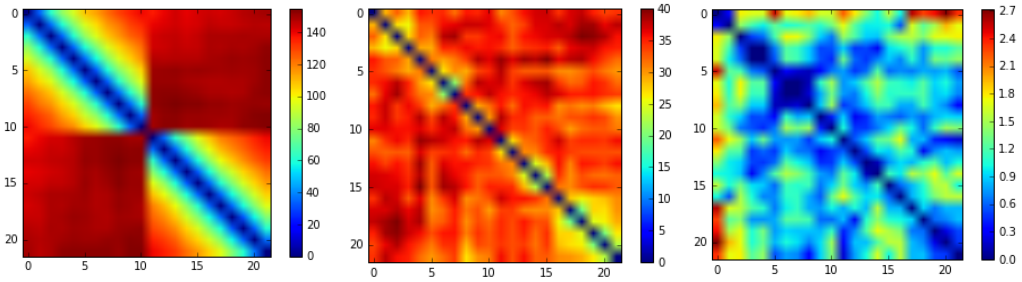


Figure 2: Hamming distance

Figure 3: Evaluation of distances for 400 edge swaps.

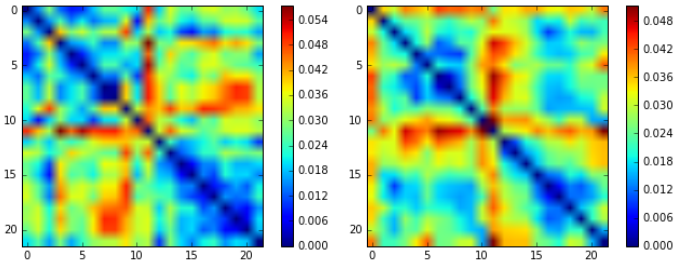
a graph obtained with another DD. We do the same experiment with 1000 swaps to show that we saturates after a certain number of swaps (the number of swaps of saturation depends on the number of the nodes, and on the type of the graph). This is shown in Figure 5. This proves that with the Edge swaps techniques, we can obtain graphs that are significantly different, but we didn't develop a method to estimate the number of edges swaps needed to obtain a significantly different graph in the general case.



(a) Hamming distance

(b) Dijkstra histograms distance.

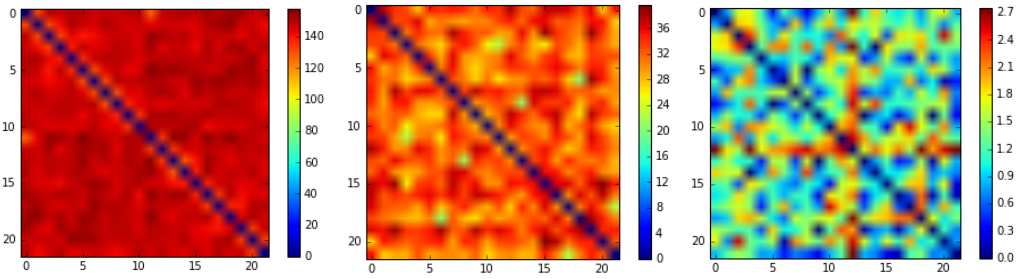
(c) Page rank histograms distance.



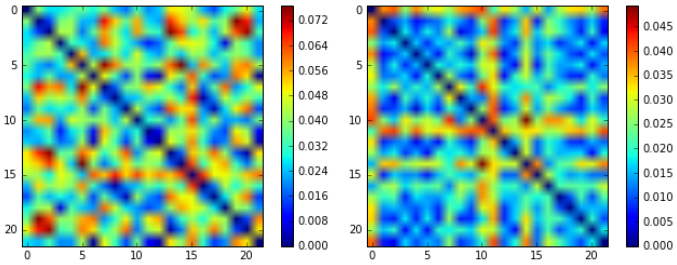
(d) Average of the max eigenvector distance.

(e) Degree correlation distance.

Figure 4: Evaluation of distances for 100 edge swaps.



(a) Hamming distance (b) Dijkstra histograms distance. (c) Page rank histograms distance.



(d) Average of the max eigenvector distance. (e) Degree correlation distance.

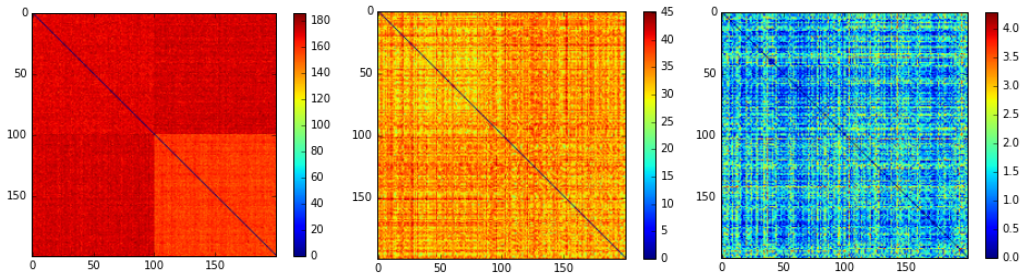
Figure 5: Evaluation of distances for 1000 edge swaps.

We now compare the variance of the graphs generated with the methods presented in the first section : Havel-Hakimi with swaps (MC sampling) and the independent sampling. For the configuration model the repeat method takes an unacceptable long time to generate graphs and the graphs obtained by the erase method do not respect the exact degree sequence and thus can be very dissimilar to the other graphs generated with the other methods. In figure 6, we show the distance matrices between different graphs generated with the same DD with the two different techniques. The top left square represent the distances between images obtained with the edge swap method (with 200swap for a 50 vertex graph, the bottom right square contains the distances between graphs generated with the independent sampling technique and the other 2 squares contains inter-distances. The results are averaged on 50 samples. The hamming distances shows a higher variance for graphs obtained with the edge swap technique, this can be due to the fact that the last technique doesn't sample uniformly the graphs and that a certain class of graph is dominant .However the Dijkstra and the degree correlation distances shows a higher variance in the generated graph for the other method. Finally, the type of the graphs obtained by the two techniques seems to be quite different from each others as we have a higher distance (top right corner and the bottom left corner)

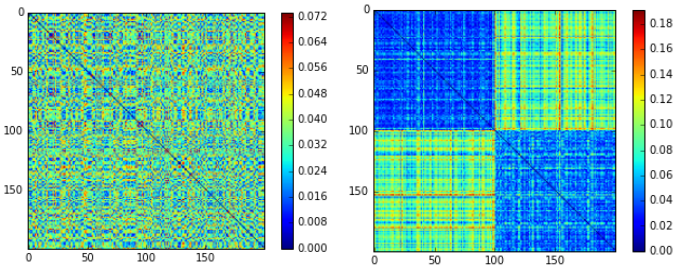
3.3 Variance of the graphs for different types of graphs

Finally, we have tried to evaluate how the number of the possible significantly different generated graph vary as a function of the type of the graph. We try to evaluate this by fixing a number of vertexes, generating a number of graphs and evaluating the pair-wise distances between the graphs. If for a certain method we can generate more different graphs, the value on the distances matrix should be higher. We can compare also the histograms. We have evaluated this on graphs of 10 nodes, by generating 100 different graph with each technique. (We need a much higher number of graphs to have significant results with more nodes, which requires more computation time and memory). We show the result obtained for the small word, preferential attachment and Erdos-Renyi graphs in Figure 7. In this particular case, we obtain more different graphs with small world than Erdos-Renyi than preferential attachment. However these results vary depending on the used metrics and the parameter with which we construct the graphs and we can not make any concusion without making more experiments with a better way to evaluate the graph differences.

The question of enumerating the number of significantly different graph is more complicated. The exhaustive search for all graph combination can be very expensive. We can also try another methods

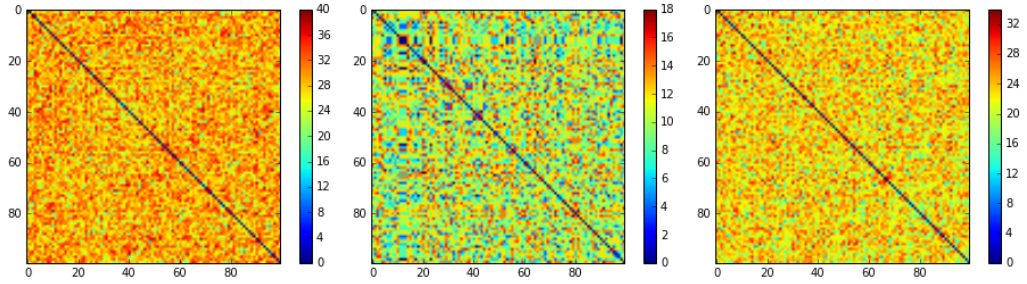


(a) Hamming distance (b) Dijkstra histograms distance. (c) Page rank histograms distance.



(d) Average of the max eigenvector distance. (e) Degree correlation distance.

Figure 6: Evaluation of the variance in the different graph generation techniques.



(a) Hamming distance for small world graph. (b) Hamming distance for preferential attachment graph. (c) Hamming distance for Erdos-Renyi graph.

Figure 7: Evaluation of the variance in the different graph generation techniques.

based on estimating the number of different graphs as a function of the number of independently sampled graphs, and extrapolate the result to an infinite number of samples, but we didn't try to do it in this project.

4 Conclusion

In this project, we have presented various methods for generating directed graphs giving a degree sequence. All of the studied methods comes with their flow. For the Havel-Hakimi with edges swap, we don't know how many edges swap we need to have an independent sampling and the method is not very efficient. The configuration model leads to many rejection and we need to use approximation in order to be efficient. The last method still has a high complexity and doesn't sample uniformly, even if it can compute weights proportional to the graph sampling probabilities. We have also taken a look at how different are the graphs that we generate with these methods. We can observe some results on the number of the swaps needed to obtain a significantly different graph or to compare the difference between graphs generated by the different methods. However those results depends on the similarity measures that we are using to find the significantly different graphs as this question is still not completely solved. Finally, we have tried to see if we can find a relation

between the types of the graphs and the number of significantly different graphs that we can obtain, but we could only give results on particular cases and no general conclusions.

References

[Hav55] Václav Havel. "A remark on the existence of finite graphs". In: *Casopis Pest. Mat.* 80 (1955), pp. 477–480.

[Hak62] S Hakami. "On the realizability of a set of integers as degrees of the vertices of a graph". In: *SIAM Journal Applied Mathematics* (1962).

[WF94] S. Wasserman and K. Faust. "Social Network Analysis". In: *Cambridge University Press* (1994).

[New02] M. E. J. Newman. "Assortative Mixing in Networks". In: *Phys. Rev.* Lett. 89 (2002).

[EMT10] Péter L Erdos, István Miklós, and Zoltán Toroczkai. "A simple Havel-Hakimi type algorithm to realize graphical degree sequences of directed graphs". In: *Electronic Journal of Combinatorics* 17.1 (2010), R66.

[BW12] Gergana Bounova and Olivier L. de Weck. "Overview of metrics and their correlation patterns for multiple-metric topology analysis on heterogeneous graph ensembles". In: *Physical Review E* 85 (2012).

[Kim+12] Hyunju Kim et al. "Constructing and sampling directed graphs with given degree sequences". In: *New Journal of Physics* 14.2 (2012), p. 023012.

[Olv12] Mariana Olvera-Cravioto. "Directed Random Graphs with Given Degree Distributions". In: *Columbia University* (2012).