Course Project Report for
CPSC-629 Spring 2014

Chaofan Li UIN:923002382
li3939108@tamu.edu

1 Degree-Balanced Random Graph Generation

Firstly, I define a term *Degeree-Balanced Graph*, which is what we want to generate.

Degeree-Balanced Graph A Degree-Balanced Graph (DBG) is an undirected graph $G(V, E)$ in which every vertex has the same degree D. For any such graph G, we say G is *degree-balanced* and D is the *degree* of G.

Let $|V|$ be the total number of vertices and D the exact degree that every vertex has in a Graph $G(V, E)$. Then, the following two statements must be true, if G is a DBG, and there must exist a DBG among all the graphs with these statements satisfied.

$$|V| - 1 \geqslant D \tag{1}$$

$$D \times |V| \mod 2 \equiv 0 \tag{2}$$

It is simple to see that if either of the two statements is not true, the graph cannot be a DBG. First consider the complete graph. For a complete graph, it has maximal number of edges, which is $\frac{|V|(|V|-1)}{2}$, among all the graphs with |V| edges. A complete graph also has maximal number of total degrees, which is exactly $|V|(|V| - 1)$, because every edge generates exactly two degrees. If a complete graph with $|V|$ vertices cannot even generates as many degrees as a DBG with the same number of vertices needs, neither can other graphs with $|V|$ vertices. Hence, $|V|(|V| - 1) \geqslant D|V|$, i.e., $|V| - 1 \geqslant D$.

As I have mentioned before, every edge generates exactly two degrees, which means that the number of total degrees must be an even number. So we have the second statement. In the remaining part of this report, I will propose an algorithm to generate a random DBG with $|V|$ vertices and the above two statements satisfied, and then we can see the existence of DBG among such graphs.

1.1 Random DBG Generation algorithm

We can generate a random graph with $|V|$ vertices and $|E|$ edges, with no other constraints, simply by selecting $|E|$ pairs of vertices randomly. However, this method doesn't work for random DBG generation, because it may generate a graph that is not degree-balanced. Here I propose an algorithm that can generate a degree-D DBG with $|V|$ vertices, as shown in algorithm 1.

input : The number of vertices $|V|$ and the degree D of the random DBG to be generated
output: A random DBG

```
1  begin
2      for i from 0 to D do // Create D + 1 empty sets, to store vertices with corresponding degree
3          sets[i] ← [ ]
4      end
5      for i from 0 to |V| - 1 do            // Create |V| 0-degree vertices and store them in sets[0]
6          sets[0][i] ← new_vertex(i + 1)                    // the label of the new vertex is i + 1
7      end
8      while length(sets[D]) < |V| - 1 do
9          v1 ← pop(random_vertex(sets[0 to D − 1]))     // Extract a vertex that is not in sets[D]
10         l ← D − degree(v1)
11         for i from 1 to l do
12             v2 ← pop(random_min_degree_vertex(sets))
               // Extract a random vertex with min degree in sets and not connected with v1
13             connect(v1, v2)                                        // Degree of v2 increase by 1
14             add(v2, sets[degree(v2)])
15         end
16         add(v1, sets[D])
17     end
18     return Graph(sets[D])        // sets[D] will be an adjacency list, create a graph with it
19 end
```

Algorithm 1: Random DBG Generation

Chaofan Li UIN:923002382
li3939108@tamu.edu

Generally, this algorithm works by iteratively randomly selecting a pivot vertex from all vertices with degree less than D and then randomly connecting several other vertices with min total number of degrees with the pivot vertex. For example, if we want to generate a 6-degree DBG and we have selected a vertex with degree one, then we will connect this vertex with random five vertices in $sets[0]$. If $sets[0]$ has less vertices then choose remaining vertices in $sets[1]$ or $sets[2]$ if vertices in $sets[1]$ are not enough either.

To prove the correctness of this algorithm, we can firstly prove several properties of the degree sets, $sets$, when this algorithm is processing on it.

1. In every **while** loop of line 8, every vertex either climbs to the highest $sets[D]$ (the pivot vertex), or climbs only one degree (like from $sets[d]$ to $sets[d+1]$), or both (from $sets[D-1]$ to $sets[D]$).

2. Every two vertices in $sets[0 \text{ to } D-1]$ cannot be connected.

   Proof. If two of them are connected, one must be a pivot vertex, and then after the **for** loop in line 11, the pivot vertex must be in $sets[D]$, which is against that two of them are both in $sets[0 \text{ to } D-1]$. □

3. At most two of $sets[0 \text{ to } D-1]$ are not empty, and they must have consecutive indices. For example, there cannot be two non-empty sets like $sets[3]$ and $sets[1]$, if $D > 3$.

   Proof. This property can be proved inductively. We know that initially there is only $sets[0]$ non-empty, the property satisfied. Suppose the property satisfied after one **while** loop in line 8. In one more such **while** loop, if the l in line 10 is larger than the size of the lower set, all the vertices in the lower set will climb to the higher set (the lower set emptied) and some vertices in the higher set will climb even one degree higher. If the l equals the size of the lower set, then the lower set will be emptied, all the vertices in it will climb to the higher set and there will be only one set non-empty. If the l is smaller than the size of the lower set, some vertices in it will climb to the higher set. All of the cases mentioned won't generate more than two non-empty sets or two non-empty sets not consecutive. The property is also satisfied after one more **while** loop in line 8. □

After proving the above properties, the correctness can be proved.

1.1.1 Proof of the Correctness of the algorithm

Prove it inductively. First, I define a term *Reachable*.

Reachable If connecting every pair of vertices which are not in $sets[D]$ will generate more degrees than what they lack to climb to the $sets[D]$ and the total degrees they lack are even. Then we say that degree D is *reachable* for the vertices in $sets[0 \text{ to } D-1]$. More accurately, suppose the total number of vertices which are not in $sets[D]$ is $|V|$, and D minus the degrees of the two non-empty $sets$ (demended by property 3 as I have proved) are k and $k+1$ respectively and there are n and $|V|-n$ vertices in them respectively. Then if the following conditions satisfied, the $|V|$ vertices are D-degree reachable.

$$|V|(|V|-1) \geqslant (k+1)|V| - n \tag{3}$$

$$\Leftrightarrow |V| - 1 \geqslant k + 1 - \frac{n}{|V|}$$

$$\Leftrightarrow |V| - 1 \geqslant k + 1 \tag{4}$$

$$(k+1)|V| - n \mod 2 \equiv 0 \tag{5}$$

Chaofan Li UIN:923002382
li3939108@tamu.edu

We can see that if the initialized $sets[0]$ satisfys the two statements mentioned at the beginning, which is just a special case of the two conditions above, i.e., $k+1=D$ and $n=0$, then they are D-reachable.

Then we can prove it inductively, if $|V|$ vertices not in $sets[D]$ are D-reachable, then as decided by the algorithm, if only the pivot vertex added to the $sets[D]$, after a **while** loop in line 8, the remaining $|V|-1$ vertices not in $sets[D]$ must be also D-reachable.

The inductiveness of (5) is obvious, because every loop will only decrese even number of lacking degrees (every edge generates two degrees).So we only need to prove the inductiveness of (3). As we can see, there are at least $2k$ decreasing lacking degrees after one **while** loop.

$$\begin{aligned}(|V|-1)(|V|-2) &\geqslant (k+1)(|V|-2)\\ &\geqslant (k+1)(|V|-2)-(n-2) \text{ when } n\geqslant 2\\ &= (k+1)|V|-n-2k \end{aligned} \tag{6}$$

If $n=0$, there are at least $2k+2$ decreasing lacking degrees. (6) should be $(k+1)|V|-n-(2k+2)$, which equals $(k+1)(|V|-2)$. So it is true for case of $n=0$.

If $n=1$, only when $|V|-1=k+1$, (6) will be unsatisfied, but when $|V|-1=k+1$, (5) is $|V|(|V|-1)-1$ which must be odd and thus has been unsatisfied.

So if only one vertex added to the $sets[D]$, the reachableness of before the loop implies the reachableness after the loop.

If not just the pivot vertex added to the $sets[D]$, suppose previously the vertices are D-reachable, after this loop, there will be only the $sets[D-1]$ non-empty. Then only condition (5) need to be satisfied, which must be true. So if initially the conditions satisfied, i.e., (1) and (2) satisfied, there always be a DBG we can generate with the algorithm I have proposed.