Network Science

Kexing Ying

May 15, 2020

Contents

1 Fundamental Graph Theory **1**
1.1 Basic Concepts 1
1.2 Graph as Models 2
1.3 Graph Isomorphism 3

2 Trees **3**
2.1 Basic Properties of Trees 3

3 Working with Networks **6**
3.1 Degree Distribution 6
3.2 Path & Distance 7

1 Fundamental Graph Theory

1.1 Basic Concepts

We begin with a few basic definitions.

Definition 1.1 (Graph)**.** A graph G is a tuple $(V(G), E(G))$ equipped with a function $\sim : E(G) \to V(G) \to V(G) \to \text{Prop}$ where $V(G)$ is the *vertex set*, $E(G)$ is the *edge set* and for all $e \in E(G)$ there exists a unique pair $v_1, v_2 \in V(G)$ such that $v_1 \sim_e v_2$. We write $v_1 \sim_e v_2$ as a short hand for $\sim (e, v_1, v_2) = \text{true}$.

Note that this definition works for both directed and undirected graphs as by this definition, a undirected graph is a directed graph with the condition that for all $e \in E(G)$, $\sim_e$ is symmetric.

Definition 1.2 (Subgraph)**.** Let G be a graph, then H is a subgraph of G if and only if H is a graph such that $V(H) \subseteq V(G)$, $E(H) \subseteq E(G)$ and the restriction $\sim^G|_H = \sim^H$. We will write $H \leq G$ for H is a subgraph of G.

Definition 1.3 (Loop)**.** Let $G := (V(G), E(G))$ be a graph and $e \in E(G)$ be an edge. We say e is a loop at some $v \in V(G)$ if and only if $v \sim_e v$.

Definition 1.4 (Multiple Edges)**.** Let $G := (V(G), E(G))$ be a graph and $e, f \in E(G)$ be edges. We call e, f be multiple edges if and only if there exists $v_1, v_2 \in V(G)$ such that $v_1 \sim_e v_2$ and $v_1 \sim_f v_2$.

Definition 1.5 (Simple)**.** We call a graph simple if it contains no loops nor multiple edges.

If a graph is simple we can then model the edge set of the graph $E(G)$ by a set of unordered tuples where each edge e with end points v_1, v_2 can be uniquely represented by $e = v_1v_2$ (commutative if and only if G is undirected).

Definition 1.6 (Complete Graph)**.** A graph G is complete if and only if G is simple and every vertex is adjacent to every other vertices, i.e. $E(G) = \{v_iv_j \mid v_i, v_j \in V(G), i \neq j\}$.

Definition 1.7 (Adjacent)**.** Let $G := (V(G), E(G))$ be a graph and $v_1, v_2 \in V(G)$, then v_1 and v_2 are adjacent (or are neighbours) if and only if there exists some edge $e \in E(G)$ such that $v_1 \sim_e v_2$.

Definition 1.8 (Path)**.** Let G be a graph, then a path in G is a simple subgraph P of G such that $V(P)$ can be ordered in a list such that consecutive vertices are adjacent. On top of this, if this ordering resulted in the first element to be adjacent to the last, then we say P is a *cycle*.

Note that we said *ordered*, so each element of $V(P)$ can only appear once in the arrangement (so no infinite loops back and forth). This can be somewhat limiting as sometimes lists with duplicate naturally arises, so let us consider the path induced by such a list.

Lemma 1. *Let G be a graph, $v_1, v_2 \in V(G)$, and L a finite list of vertices of G (possibly with repeats) such that consecutive vertices are adjacent. Then v_1, v_2 are connected.*

Proof. We present an algorithm to find such a path. While L contains duplicates, find the first pair of duplicates and remove every vertex in the sequence between the two duplicates including the last duplicate. This algorithm will always terminate as L is finite. □

Definition 1.9 (Connected)**.** A graph G is connected if and only if for all $u, v \in V(G)$, there exists a path P in G such that the rearrangement of P begins in u and ends in v.

1.2 Graph as Models

Definition 1.10 (Complement)**.** Let G be a simple graph, then the complement of G, $\bar{G}$ is the simple graph $(V(G), E(\bar{G}))$ where for all $u, v \in V(G)$, $uv \in E(\bar{G})$ if and only if $uv \notin E(G)$.

Note that this complement graph is unique only if we restrict it to be simple. Suppose G is simple and let $v \in V(G)$, then $vv \notin E(G)$ by the no loop condition. Thus, if we do not restrict $\bar{G}$ to be simple, then we can add how many loops as we want at v, making the complement not unique.

Proposition 0.1. *Let G be a simple graph, then the complement of G is unique.*

Proof. Let G_1, G_2 be complements of G. By definition $V(G_1) = V(G) = V(G_2)$ so $G_1 = G_2$ if and only if $E(G_1) = E(G_2)$. Wlog. it suffices to show that $E(G_1) \subseteq E(G_2)$. Let $uv \in E(G_1)$, then $uv \notin E(G)$ and thus $uv \in E(G_2)$. □

Let us consider a real world problem. Suppose we have n job openings and k applicants but not all applicants are qualified for all jobs. We can easily model this problem by connecting each applicants to their respective qualified jobs and ask whether we can find a subgraph that consist of n pairwise disjoint edges.

Upon examining this question, we find that this particular model has an interesting graph structure in which none of the jobs are adjacent to each other (similarly for the applicants). This type of graphs are called *bipartite* and the set vertices representing people and jobs respectively are called independent.

Definition 1.11 (Independent)**.** Let G be a graph and $S \subseteq V(G)$. S is called an independent set in G if and only if for all $u, v \in S$, $uv \notin E(G)$.

Definition 1.12 (Bipartite)**.** A graph G is called bipartite if and only if $V(G)$ is the disjoint union of two independent sets in G. We call these two independent sets the *partite* sets of G.

1.3 Graph Isomorphism

Similar to all mathematical structures, there exists a notion of isomorphisms between two graphs that are essentially the same[1]

Definition 1.13 (Isomorphism)**.** An isomorphism from a simple graph G to a simple graph H is a bijection $f : V(G) \to V(H)$ such that for all $uv \in E(G)$, $f(u)f(v) \in E(H)$. We write $G \cong H$ as usual.

Theorem 1. *Graph isomorphisms is a equivalence relation.*

As usual, theorems of the above kind is simply by checking each of the properties so we will omit it here. As an equivalence relation induces a partition, a set of graphs can be quotiented out by the isomorphism relation and we call each element of this quotient an *isomorphism class*.

As all graphs in a isomorphism class are pairwise isomorphic to one another, they all share graph structures. Therefore, when discussing graph structures, it makes sense to talk about a isomorphism class rather than a particular graph. When we do this, we will informally call it an *unlabeled* graph.

We will now introduce some notations. The unlabeled path and cycle of n vertices is denoted by P_n and C_n respectively while the complete graph of n vertices is denoted by K_n. If G is a bipartite graph with n vertices such that two vertices are adjacent if and only if they are in different partite sets, then we call G a complete bipartite graph and denote it by $K_{r,s}$ where r, s are the sizes of the partite sets.

2 Trees

2.1 Basic Properties of Trees

Definition 2.1 (Minimally Connected)**.** We say a graph G is minimally connected if and only if it is connected and for all $e \in E(G)$, the graph $(V(G), E(G) \setminus \{e\})$ is not connected.

[1] By essentially the same we mean that almost all properties of a graph commutes via an isomorphism.

Definition 2.2 (Tree)**.** A tree is a minimally connected graph.

Lemma 2. *A tree does not have any cycles.*

Proof. Let G be a tree and for contradiction suppose $C \leq G$ is a cycle. Then by the definition of a cycle, there exists $v \in V(C)$, such that elements of $V(C)$ can be arranged into a circle of consecutively adjacent vertices. Let u, v be the first and last element of $V(C)$ in the above arrangement, and $x, y \in V(G')$ where $(G' := V(G), E(G) \setminus \{uv\})$. As the there path arranged above remains, it follows that u and v are connected. So, as $x, y \in V(G') = V(G)$, there exists some path P from x to y in G. Now, if $uv \in E(P)$, we can simply replace that with path from u to v implying G' is also connected. # □

Lemma 3. *The graph G is a tree if and only if it is connected and acyclic.*

Proof. The forward direction follows directly from the above lemma so it suffices to show that G is minimally connected if it it connected and acyclic.

Suppose that G is not minimally connected, then there exists $uv \in E(G)$ such that $G' := (V(G), E(G) \setminus \{uv\})$ is also connected, so there exists a path P in G' from u to v. Now, as this sequence P by definition begins with u and end with v, and u, v are adjacent by uv in G, this sequence is therefore a cycle in G. # □

Lemma 4. *Any two vertices on a tree is joined uniquely by a path.*

Proof. Existence is true by the connectedness of a tree so all that remains is to prove that the path is unique.

Let T be a tree, $x, y \in V(T)$, and P_1, P_2 paths from x to y in T. Suppose $P_1 \neq P_2$ and let us denote p_1, p_2 the path arrangement. Let $S := \{n \in \mathbb{N} \mid p_1[n] \neq p_2[n]\}$, then, as $P_1 \neq P_2$, S is not empty; and the fact that paths are finite, S has a minimum and a maximum value i, j by the well-ordering principle. Now, by connecting $p_1[i-1 : j+1]$ to $p_2[i-1 : j+1]$ (both of which index makes sense as $i > 1$ as $p_1[1] = x = p_2[1]$ and similarly for j), and by considering lemma 1, we have created a cycle in T. # □

In nature, many trees have *leaves* and therefore, so do our graph-theoretic trees (in fact our trees are stronger as *all* trees with two or more vertices have at least two leaves).

Definition 2.3 (Leaf)**.** A vertex in a tree is a leaf if and only if it has degree one, i.e. it is only connected to one other vertex.

Lemma 5. *A tree with two or more vertices has at least two leaves.*

Proof. Consider the longest path in this tree. It must have a leaf at either end of the path as otherwise it is not the longest path in this tree. □

Lemma 6. *Let T be a tree and $v \in V(T)$ a leaf. Then $T - v$ is also a tree*[2]*.*

[2]We write $T - v$ for the graph $(V(T) \setminus \{v\}, E(T) \setminus S)$ where S is the set of edges with v as an endpoint.

Proof. $T - v$ is connected as for all $x, y \in V(T - v)$, x, y is connected in T by some path P. Now, as $v \notin V(P)$ since if otherwise v must be on the either end of the path implying either x or y equals v. Also, as $v \notin V(P)$, e_v, the unique edge with endpoint v is also not in P, so $P \leq T - v$ connecting x, and y.

Thus, by lemma 3, it suffices to show that $T - v$ is acyclic. This is trivial as if $C \leq T - v$ is a cycle, then it is also a cycle in T, so we are done! □

Lemma 7. *A tree of n vertices has $n - 1$ edges.*

Proof. For convenience let us denote the trees of n vertices by T_n.

We apply natural number's induction on n. If $n = 1$ the result is trivial. Suppose for $n = k$, for all T_k, $|E(T_k)| = k - 1$, and let us consider the case for the tree T with $n = k + 1$ vertices.

By lemma 5, T has at least two leaves and let l be one of the leaves. Then by lemma 6, $T - l$ is a tree of n vertices, so it has $k - 1$ number of edges. However, as l is a leaf, there is only one edge with endpoint l, so $|E(T)| = |E(T - v)| + 1 = k - 1 + 1 = k$. □

Lemma 8. *A connected graph G of n vertices has at least $n - 1$ edges.*

Proof. We induct on the number of vertices. As the number of edges of a graph is a natural number, it is greater or equal to $0 = 1 - 1$, so true for $n = 1$.

Suppose G has $k + 1$ vertices, then by excluded middle, either G is minimally connected, or it is not. If it is, then it has $n - 1$ edges by the previous lemma so suppose otherwise. Then there exists $v \in V(G)$ such that $G - v$ is still connected. As $G - v$ has k vertices, it has at least $k - 1$ number of edges by the inductive hypothesis. Now, as G is connected, v is at least connected to another vertex, thus, G has at least k edges. □

Lemma 9. *A connected graph G of n vertices is a tree if and only if it has $n - 1$ edges.*

Proof. The forward direction of the proof is exact lemma 7 so let us consider the reverse direction.

As G is connected, by lemma 3, it suffices to show that G is not cyclic. In order to show this, we apply induction on the number of vertices of G.

If G has one vertex, then it is trivially acyclic. Now suppose for connected graphs G_k of k vertices and $k - 1$ edges are trees, let us consider the connected graph of G with $k + 1$ vertices and k edges. If G is minimally connected then G is acyclic by lemma 3 so suppose that there exists $v \in V(G)$ such that $G - v$ is connected. Now as G is connected, there exists at least one edge who has endpoint v, so $G - v$ has at most $k - 1$ edges. However, as $G - v$ is connected, it has at least $k - 1$ edges by the above lemma so it has exactly $k - 1$ edges and v is a leaf. Therefore, as $G - v$ is a tree by the inductive hypothesis, we have G is also a tree. □

3 Working with Networks

In this section we will be less rigorous and focus more on the methods used to analysis graphs (networks) especially really large ones.

3.1 Degree Distribution

Network systems vary in size but they are normally very large (that is they are large enough such that we can't draw them by hand), so, in order to analyse large networks, it is often useful to take a probabilistic approach.

Definition 3.1 (Degree of a Vertex of a Undirected Graph)**.** Let $(V(G), E(G))$ be a undirected graph and $v \in V(G)$, if v is the end point of some $e \in E(G)$, then we say v and e are *incident*. Then the degree of v is the number of incident edges.

Suppose we denote the degree of some vertex v by $d(v)$, then we find the total number of edges is simply

$$|E(G)| = \frac{1}{2} \sum_{v \in V(G)} d(v).$$

Note that the 1/2 factor is because each edge is incident to two vertices.

Definition 3.2 (Average Degree of a Undirected Graph)**.** Let $G = (V(G), E(G))$ be a undirected graph, then the average degree of G is $\frac{1}{|V(G)|} \sum_{v \in V(G)} d(v) = 2\,|E(G)| \,/\, |V(G)|$.

The above, however, does not simply transfer to directed graphs since we would loss the information of "directedness" of the graph. Therefore, the degree is defined slightly differently for directed graphs.

Definition 3.3 (Degree of a Vertex of a Directed Graph)**.** Let $(V(G), E(G))$ be a directed graph and $v \in V(G)$, then the degree of v is simply the difference between number incoming edges and the number of out going edges.

With the definition above, we see straight away the sum of the degrees of all vertices in a directed graph is zero so the definition for average degree does not apply for digraphs[3] either. Therefore, instead defining the average degree by averaging the sum of degrees, we use the average of the sum of either the incoming or outgoing degrees (both of which are equal).

Let us now consider the *degree distribution*, p_k, a characterisation of a graph that provides the probability that a randomly selected vertex has k degree.

Straight away, given some graph $(V(G), E(G))$, let $S := \{v \in V(G) \mid d(v) = k\}$, then $p_k = |S| \,/\, |V(G)|$.

[3]Digraph is an alternative word to directed graph.

3.1.1 Clustering Coefficient

The clustering coefficient of a graph attempts to capture the degree to which the adjacent nodes of a given node is connected to each other.

Definition 3.4 (Clustering Coefficent)**.** Let G be a graph and $u \in V(G)$ is a node with degree $d(u)$, then the clustering coefficient of u is

$$C_u = \frac{2L(u)}{d(u)(d(u)-1)},$$

where $L(u)$ is the number of edges in $E(G)$ connecting two neighbours of u.

We note that the clustering coefficient C_u is a number between 0 and 1 with $C_u = 0$ representing none of the adjacent nodes of u is adjacent with each other which $C_u = 1$ means every adjacent node of u is adjacent to every other adjacent node of u.

If the graph in question is *simple*, then we see that the clustering coefficient of some node u represents the probability that two (unique) adjacent nodes of u are adjacent[4].

Theorem 2. *An acyclic graph has zero clustering coefficients everywhere.*

Proof. This follows directly from that if there exists some node with clustering coefficient greater than 0, it has at least two neighbouring nodes which are adjacent to each other. Then we can easily create a path starting and ending at this node creating a cycle. # □

Corollary 2.1. *A tree has zero clustering coefficients everywhere.*

By taking the average of the clustering coefficient of every node, we have the *average clustering coefficient* which represents the degree of clustering over the whole network.

Definition 3.5 (Average Clustering Coefficien)**.** Let G be a graph and for all $u \in V(G)$ we denoted the clustering coefficient of u by C_u, then the average clustering coefficient of G is

$$\langle C \rangle = \frac{1}{|V(G)|} \sum_{u \in V(G)} C_u.$$

3.2 Path & Distance

Given a graph G and two nodes $u, v \in V(G)$, we would often like to quantitatively construct an ordering on these paths. This can be achieved through creating a distance function on the set of paths.

Definition 3.6 (Distance of Unweighed Paths)**.** Let G be a graph with unweighed edges and $u, v \in V(G)$. Suppose that P is a path between u and v in G, then the distance of P is

$$\text{dist} P = |P| - 1,$$

where $|P|$ denoted the length of the sequence of nodes induced by P.

[4]This is because $d(u)(d(u)-1)$ is the number of ordered combinations of choosing two nodes while for each edge counted by $L(u)$, it connects two unique parings (u, v) and (v, u).

It is not hard to imagine why such an distance function is useful, and why often we might want to look for the shortest path between to nodes in a network.

Definition 3.7 (Shortest Path)**.** Let G be a graph and $u, v \in V(G)$. Suppose we denote S as the set of paths between u and v. Then, we call $P \in S$ a shortest path if and only if

$$\text{dist}P = \min_{Q \in S} \text{dist}Q.$$

We denote this distance by $d_{u,v}$.

This definition makes sense as we are work with finite networks, and thus, the number of paths between two nodes is finite. We note that the shortest path between two nodes is not unique as many paths can have the same distance.

We quickly introduce some more definitions. Given a graph G, and a path P in G,

- the *diameter* of G is $\max\{d_{u,v} \mid u, v \in V(G)\}$;
- the *average path length* of G is $\frac{1}{|S|}\sum_{d \in S} d$ where $S = \{d_{u,v} \mid u, v \in V(G)\}$;
- P is an *Eulerian path* if and only if $E(G) \subseteq P$;
- P is a *Hamiltonian path* if and only if for all $u \in V(G)$, there exists $e \in P$, u is an endpoint of e.

3.2.1 Breadth-First Search (BFS) Algorithm

The BFS algorithm is an algorithm to determine the shortest distance between two nodes of a network.

Let G be a network and $u, v \in V(G)$, then

1. For each adjacent nodes w of u, construct a tuple $(w, d(w) = 1)$ and put them in a queue.
2. While the first element w of the queue is not equal to v, for each adjacent node x of w, construct a tuple $(x, d(x) = d(w) + 1)$ and remove w from the queue.
3. If the first element of the queue is v, then the distance is simply $d(v)$. However, if there is no element remain in the queue, we know that the two nodes are not connected.

Note that we can optimize the algorithm by making sure that we do not go back to previously visited nodes but that's simply an implementation issue on how one can achieve this.

We see that the complexity of this algorithm is $O(|V|)$ since every node will be explored in the worst case.