Discovering Locally Dense Groups

by chew lu 11/05/2018

What is Groups, What is Locally Dense Groups?

Finding groups in a network is of interest, and we need to know what characteristics do the groups we intent to find have. The most common and necessary property a group should have is cohesiveness and here the group we talk about is cohesive group. There is four characteristics of cohesiveness:

1. Mutuality.
2. Density.
3. Compactness.
4. Separation.

They are all representing the cohesiveness of a network with different emphasis, and here we will focus on the density, with is the fundamental and important. First, it is observed that the assortative mixing is a common property in a social network group, which means that the members intend to have the properties that their neighbors have. When the property is the degree, then, the density matters (we use degree to compute the density, so as we can use density to imply the degree). Second, it is mathematically proved that high density implies other cohesiveness characteristics.

And the term "locally" means the invariant under network changes outside the group. A local property is definable over all the vertexes inside the group. There are non-local dense group (subgraph or subset of V) such as LS Sets, Lambda Sets and Normal Set, and they all need to compare to the network outside to determine that whether the group is "dense". (These sets concerned about the "betweenness" or "connectivity" from the group to the network outside).

And we usually have a maximality condition where we want to discover some groups. That is to find the maximum group that satisfied the query condition.

Without specially specification, we only consider the undirected unweighted graph here.

The Prototype of other kind of Group : Clique

Definitions :

Clique : a complete subgraph of a graph G.

Maximum Clique : a clique with maximum cardinality.

Maximal Clique : a clique with maximal vertexes.

N-Clique : a subgraph whose vertexes' distances are smaller than N.

N-Club : a subgraph with a diameter smaller than N.

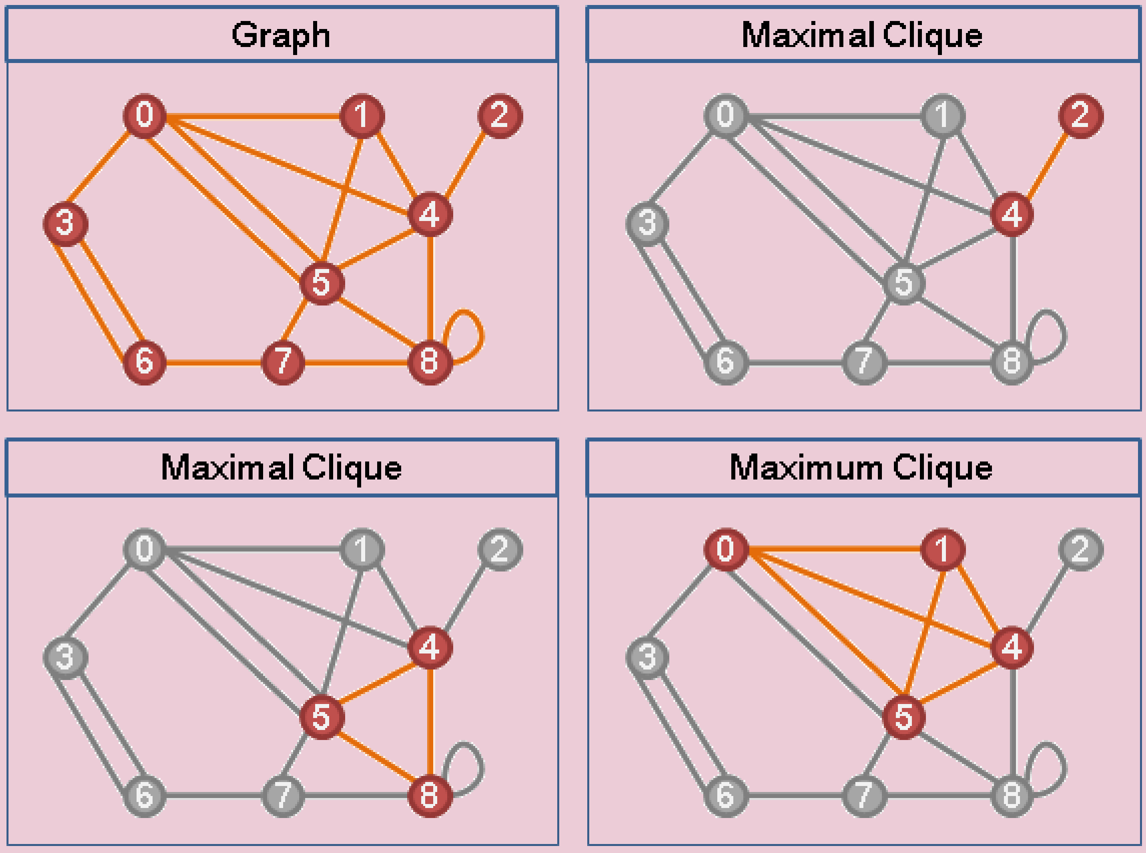
N-Clan : a subgraph is both a N-Clique and N-Club.

The last three definition are not important.

Comparing maximum clique and maximal clique:

「極大團 Maximal Clique 」是無法直接增加頂點數目的團，可能有許多個。

「最大團 Maximum Clique 」是頂點數目最多的團，可能有許多個。最大團是所有極大團當中最大的；最大團也是極大團。



Properties of Clique

1. Perfectly dense.
2. Perfectly compact.
3. Perfectly connected.
4. Closed under exclusion.
5. Nested.

Theorems

The existence of cliques of size k :

Theorem 6.1.2 (Turán, 1941). *Let $G = (V, E)$ be an undirected graph. If $m > \frac{n^2}{2} \cdot \frac{k-2}{k-1}$, then there exists a clique of size k within G.*

The unbounded number of maximal cliques :

Theorem 6.1.3 (Moon and Moser, 1965). *Every undirected graph G with n vertices has at most $3^{\lceil \frac{n}{3} \rceil}$ maximal cliques.*

A enforced theorem of 6.1.2 and this kind of research is call *extremal graphs*:

Theorem 6.3.6 (Dirac, 1963). *Let $G = (V, E)$ be any undirected graph. If $m > \frac{n^2}{2} \cdot \frac{k-2}{k-1}$, then G contains subgraphs of size $k + r$ having average degree at least $k + r - 1 - \frac{r}{k+r}$ for all $r \in \{0, \ldots, k-2\}$ and $n \geq k + r$.*

Problems

1. We can know the existence of a cliques of size k, but we have difficulty to locate them.
2. We have enormous maximal cliques and they may overlap each others, and we have troubles to count them and rank them.
3. Brute-force algorithm is computational unacceptable. Which means the tractability.

Relaxations and Variations of Clique

We will take about them later, and here we give their name and main properties :

subgroup	closed under exclusion	nested	tractable
clique	+	+	−
N-plex (for $N \in \mathbb{N}$)	+	+	−
N-core (for $N \in \mathbb{N}$)	−	−	+
η-dense subgraph (for $\eta \in [0, 1]$)	−	+	−

Finding the Size of Maximum Clique is an NP-problem But we can still do something

We translate the problem of computing the size of maximum clique into determining whether a conjunctive normal form H is satisfiable by constructing a K-partite graph such that:

$V_H = \{(L, i) | L \; is \; a \; literal \; of \; clause \; i \; in \; H\}$

$E_H = \{\{(L, i), (L', j)\} | L \neq \neg L' \; and \; i \neq j\}$

So, an edge exist if and only if two conjunctive clauses that are composed of disjunctive literals are satisfiable, then an an clique of size k exist if and only if H of k clauses is satisfiable.

Then, we gonna give an upper bound of this question/decision.

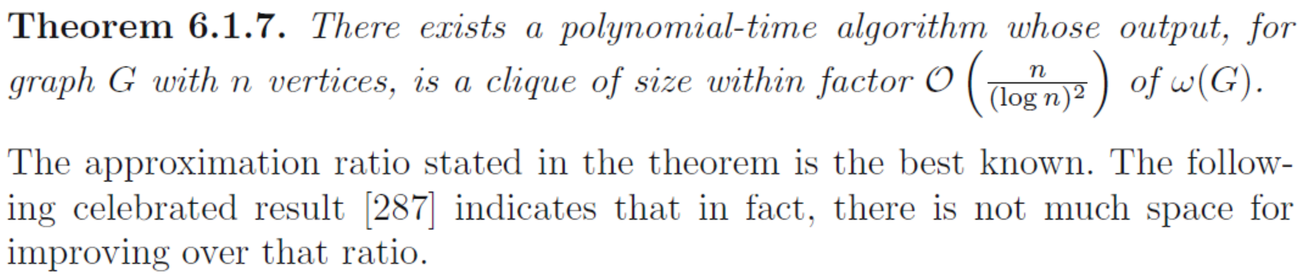
If the minimum degree of a graph is greater than n-4, than we can compute the maximum clique easily which is a bottom condition. Let's assume the minimum degree is not greater than n-4 and the vertex with the min degree is noted *v*, then the maximum clique either contain *v* or not contain *v*, and since clique is closed under exclusion then we get a recursive inequality that the running time is T(n) :

$T(n) \leq T(n-4) + T(n-1) + c \cdot (n+m)$

Solve it, then we get T(n) = O($n^2 1.3803^n$) . But the minimum upper bound remind unknown and the known fast and"space-friendly" algorithm has a running time T(n) = O($n^2 1.2227^n$) .

Then a natural idea comes out : can we get an approximate result with justified running time?

How far can we go using approximation technique or heuristic technique? Well, there is a theorem, delightful and exhausted.



Theorem 6.1.7. *There exists a polynomial-time algorithm whose output, for graph G with n vertices, is a clique of size within factor $\mathcal{O}\left(\frac{n}{(\log n)^2}\right)$ of $\omega(G)$.*

The approximation ratio stated in the theorem is the best known. The following celebrated result [287] indicates that in fact, there is not much space for improving over that ratio.

Exhaustive Searching to Find the Maximum Clique

To determent whether a give set of vertexes is a clique, the simplest way is to examine whether there is an edge between each pair. Clearly, the worst case is O(n^2).

And to exhaustively search, we need to examine every possible pair, that is to say, the total number of vertexes subsets we need to examine is the sum over $\binom{k}{n}$, which is exactly $(1+1)^n$.

Thus the worst running time is O($n^2 2^n$), which in shorthand is $O*(2^n)$.

Finding Fixed Size Clique

It should be better than finding all cliques exhaustively and then pick up the clique of certain size. And we get started from the basic problem: finding all triangles in a graph, since finding the clique of size 2 is trivial.

even the exhaustive searching can be improve when the question comes down. We just examine every subset of size k. Which takes O($k^2 n^3$).

Remark the multiplication of adjacency matrix of a graph, the result is the counting of the routes of 2 distances between each pair. Remark again that the graph is simple, undirected and unweighted and loop free.

Then, we quickly compare the result of multiplication and the original matrix, then if an index (i,j) has positive integer in both matrix, then there is a triangle. And it is faster than the exhaustive approach for the reason that the multiplication is smaller than O(n^3)

The same idea can be extended further with divide-and-conquer method and it cannot simply extended due the some facts such and the graph would not be a "loop-free" graph if a triangle exist (the same positive integer on the same position).

The algorithm to find the k-size cliques is simply described as follow:

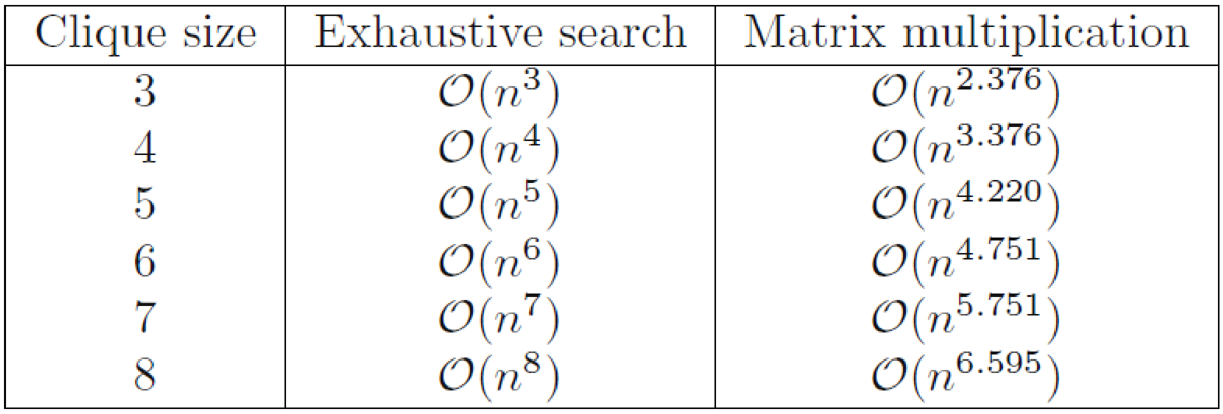
1. Find the cliques of size $\frac{k}{3}-1, \frac{k}{3}, \frac{k}{3}+1$.
2. Divided the graph into a tripartite, with three set of vertex that have all the cliques of $\frac{k}{3}-1, \frac{k}{3}, \frac{k}{3}+1$ size.
3. Create a tripartite auxiliary graph.
4. Determine whether two cliques of different partites compose a bigger clique. If so, create a two vertexes linked by a edge, and put the vertex into it corresponding partite in the auxiliary graph. All work can be done in time smaller than O($n^{\frac{2k}{3}}$)).
5. Find the triangles in the auxiliary graph, it a triangle exist, then there is a clique with size k. Finding the triangle takes three matrix multiplications, and the multiplication can be fasten to O($n^{1+\frac{2k}{3}}$} by the fast rectangular matrix multiplication. (Is it the running time is only dependent on the size of the matrix? The balanced rectangle size, we get n/(2k/3), so the running time is simple (3n/2k)^3, how can it be $O(n^{max\{k_1+k2,k_1+k_3,k_2+k_3\}})$ as the textbook described? I don't known the fast rectangular matrix multiplication, but I don't think there is a relationship between k and the multiplication running time).

Then recursive expression roughly:

T(n,k) = 3T(n ,k/3) + O($n^{\frac{2k}{3}+1}$).

So it, it is O($n^{\frac{2k}{3}+1}$). More precisely, it is $O(n^{\alpha})$, where α is the upper floor of (2k/3).

Let's compare:



Clique size	Exhaustive search	Matrix multiplication
3	$\mathcal{O}(n^3)$	$\mathcal{O}(n^{2.376})$
4	$\mathcal{O}(n^4)$	$\mathcal{O}(n^{3.376})$
5	$\mathcal{O}(n^5)$	$\mathcal{O}(n^{4.220})$
6	$\mathcal{O}(n^6)$	$\mathcal{O}(n^{4.751})$
7	$\mathcal{O}(n^7)$	$\mathcal{O}(n^{5.751})$
8	$\mathcal{O}(n^8)$	$\mathcal{O}(n^{6.595})$

Enumerating Maximal Cliques

Let's introduce the algorithm given by the textbook, which is easier to understand and a better instruction to human, comparing to the algorithm given in Wikipedia:

> Using these observations they can generate all maximal cliques in G by a recursive algorithm that chooses a vertex v arbitrarily and then, for each maximal clique K in *G \ v*, outputs both K and the clique formed by adding v to K and removing the non-neighbors of v. However, some cliques of G may be generated in this way from more than one parent clique of *G \ v*, so they eliminate duplicates by outputting a clique in G only when its parent in *G \ v* is lexicographically maximum among all possible

> parent cliques. On the basis of this principle, they show that all maximal cliques in G may be generated in time *O*(*mn*) per clique, where m is the number of edges in G and n is the number of vertices.

The textbook gives a iterative adaption of the algorithm above, it enumerates the maximal cliques by construction, ,more precisely, the construction of a binary tree. The key feature of the tree is that on every level of the tree, the node of the tree is the maximal clique at that time, at that level, at the set of the chosen vertexes from the graph.

1. Select an arbitrary vertex as its root. And this vertex form a vertex set U. the current vertex is the root.
2. Select another vertex v and then test that if the neighbors of v contains U.
    1. If so, add v to U, and make a left child on the current node.
    2. If not, there is two choices: a) move downward with v without the some vertexes in U that are not willing to go downward with v being a clique together; b) move downward with U without v, friendship left forever. The key idea is to form a clique, and to form a maximal clique. If a) works, the we make a right child on the current node, we have a new maximal clique to move forward. And we will also consider b), if b) works, old friendship goes on. Remark that, the children are the maximal cliques at the next level, only considering the set of selected vertexes.
3. repeat 2, until no vertex left.

One more thing that may be improved in the textbook is the delay between output. The book say the delay between output is $O(n^3)$, but I think it is $O((2n-1)(n+m)))$ that is $O(n^2+nm)$, but the Wikipedia tells it should be $O(mn)$:

> On the basis of this principle, they show that all maximal cliques in G may be generated in time *O*(*mn*) per clique, where m is the number of edges in G and n is the number of vertices.

There exists an algorithm to exploit the property that we can imply the second lexicographically smaller身体 clique from the first lexicographically smallest clique, as long as we know the lexicographically first clique of the graph.

Here I briefly mention the idea of the exploitation and the proof. We extract the lexicographically first clique U_0 from a minimum priority queue Q. Then find out a vertex v_j which are not adjacent to some vertex v_i which is the lexicographically preceding vertexes. Then we take the vertex v_i to form a clique T and insert T into Q. After finding out all the v_j based on U_0, then we extract a clique from Q and replace U with it and loop again until no such v_j exists which means U is either U_0 or the lexicographically last clique.

The proof is inductive, and it means to proof that the lexicographically next graph must be inserted into Q. The idea is in the reversed order of the algorithm. First, we have a clique U that is greater than U_0. Remark that the U is can be any clique that satisfied this condition. Then we can see that U will finally bring up a clique T that is smaller than U and T will be inserted into Q, which ensures the lexicographically first clique must be in Q. Second, we find out the largest index j, such that makes $U_j = \{v_1, \ldots, v_j\} \cup U$ is not a maximal clique in $\{v_1, \ldots, v_j\}$. Then, based on the construction (the maximality of j and the maximality of maximal clique), we have $v_{j+1} \in U$, and a non-empty vertex set S, such that $S \cup U_j$ is a maximal clique in $\{v_1, \ldots, v_j\}$, and $N(v_{j+1}) \cap S \neq S$ (otherwise the maximality doesn't hold). Let's note that $S \cup U_j$ is contained in a clique on V and is denoted as U_0, then U_0 is smaller than U. Then, reverse the process, let us have known U_0 first, then we get the algorithm mentioned above that produce a larger clique T that contains $U \cap \{v_1, \ldots, v_{j+1}\}$, and which is smaller than U, otherwise we would have a U_k with $j+1<k$ which is contrary to the maximality of j. Because all the generated T agrees with U_0 on $T \cap \{v_1, \ldots, vj\}$. If all v_j are considered, then the lexicographically next clique T must be in Q.

Personally, I intend to think the algorithm is inspired by someone saying that "hey, I know how to find a lexicographically smaller clique!". And then scientists pull out its potential. And, I think the algorithm for listing maximal clique in reversed lexicographical order may be easier for beginner to understand.

No polynomial time algorithm for counting the number of cliques.

Structural Relaxation of Clique : Plexe

Clique is 1-plex for the reason that the minimum degree on a vertex is 1.

Definitions

Define a minimum degree function $\delta : G \rightarrow N,\ s.t.\ \delta(G) = min\{deg(v)|v \in V\}$.

N-core is $U \in V$, such that $\delta(G[V]) \geq |U| - N$.

Properties:

1. relations among : $|U|, N, diameter(G[V])$.
    1. $N < \frac{(n+2)}{2} \rightarrow diam(G) \leq 2$, and if n > 3, then G[U] is two-edge-connected.
    2. $N \geq \frac{n+2}{2} and\ G\ is\ connected \rightarrow diam(G) \leq 2N - n + 2$. (The proof gives a useful relation between N(v) and deg(v)).
2. closed under exclusion $\rightarrow$ nested.

Does there exist an N-plex of size at least k within G?

Intuitively, this question for 1-plex is NP-complete and it seem finding N-plex is complexer than finding 1-plex, so it is NP-complete too.

A directly proof of that this question is NP-complete for all natural number N>1 is similar to the proof of that the CLIQUE problem. We prove it by construction. We translate the problem of finding clique of size k into finding an N-plex of size k+(N-1)|V|. But I did not see the reversed translation in the textbook. But it is trivial I think, because we can add vertexes and edge without decreasing the N-plex of size k+(N-1)|V| in the original graph.

The translation is describe following, to construct a graph $G'(V', E')$ from $G(V, E)$:

$V' = V \times \{0, 1, \ldots, N-1\}$, where N is the N from N-plex of size k.

$E' = E_1 \cup E_2 \cup E_3$

$E_1 = \{\{(u, 0), (v, 0)\}|(u, v \in E\}$

$E_2 = \{\{(u, i), (v, j)\}|u \neq v \in E\ and\ j > 0\}$ (Remark:the textbook allowed self-loop, but I think it is better to exclude them for preciseness and it would not affect the proof and the result)

$E_2 = \{\{(u, 0), (v, j)\}|u, v \in E\ and\ u \neq v\}$

Then, we can clique of size k in G only and only if we can find N-plex of size k+(N-1)|V| in G'.

Let's analyze first, the number K+(N-1)|V|, it is the sum of k vertexes in G and the cardinality of $\bar{V}'_0 = V' - V'_0,\ where\ V'_0 = V' - V \times \{0\}$. And we observe that, due to E_2, $\bar{V}'_0$ is a clique of size (N-1)|V|. And, if a subset U of size k in G', if we replace the vertex in $U \cap V'_0$ (if any) with a vertex from $\bar{V}'_0$, them the minimum degree of U will not decrease.

Then, we choose a subset U of size k+(N-1)|V| in G', such that $|U \cap V'_0| \geq k$. Then we iteratively replace all the vertexes in $U \cap V'_0$ with vertexes from $\bar{V}'_0$, and we finally get U'. What is the vertex v with minimum degree on U'? v' must be in V_0 and if there is a clique U_c of size k in G and due to E_1, luckily we may include U'_c in U' and have the best result that $deg(v) = k + (N-1)(|V|-1)$, which means U' is a N-plex of size K+(N-1)|V|! Let's denote the minimum degree of the vertexes for a graph G as $\delta(G)$, then we have $\delta(G'[U]) \leq \delta(G'[U']) \leq \delta(G'[U_{contains\ U'_c}]) = k + (N-1)(|V|-1)$. And the "only if" can be proved the same way (or by contradiction). Q.E.D.

Dual of N-plex : N-core

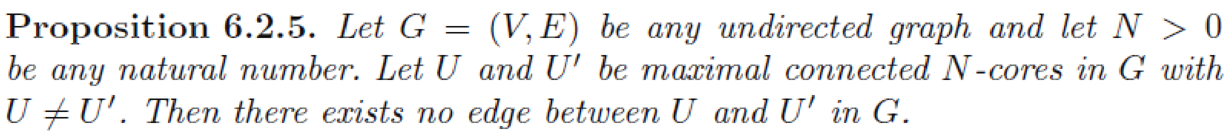
Definitions

N-core : N-core is $U \in V$, such that $\delta(G[V]) \geq N$.

core number : $\xi(v) = max\{N | v \in U\ and\ U\ is\ a\ \text{N-}core\}$

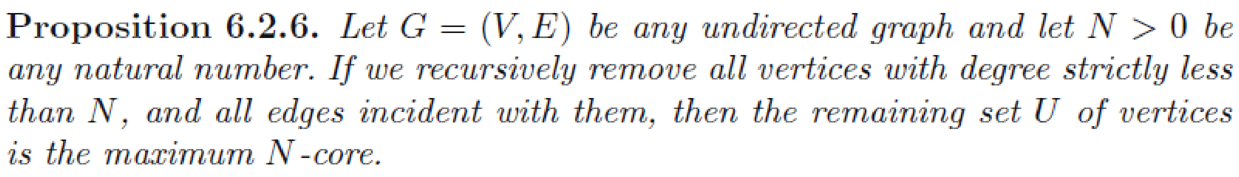
Propositions:

1. This proposition 6.2.5 can be used to determining whether a N-core is maximal. And we should be aware of the connectedness requirement for this proposition.

   Proposition 6.2.5. *Let $G = (V, E)$ be any undirected graph and let $N > 0$ be any natural number. Let U and U' be maximal connected N-cores in G with $U \neq U'$. Then there exists no edge between U and U' in G.*

2. This proposition is used to constructing maximum N-cores by which we can count the core number of each vertex. So it is also of algorithm importance.

   Proposition 6.2.6. *Let $G = (V, E)$ be any undirected graph and let $N > 0$ be any natural number. If we recursively remove all vertices with degree strictly less than N, and all edges incident with them, then the remaining set U of vertices is the maximum N-core.*

3. The closure under exclusion, the nestedness and the tractablility of N-core is dual to N-plex.

Counting the core number for each vertex

As N-core is dual to N-plex, such as, a N-plex is a (|U|-N)-core. So, finding a N-core of size k is also NP-complete.

But we can exploit the proposition 6.2.6 to compute the core number of each vertex in polynomial time.

First of all, we assume that all the vertexes have the core numbers as the same as their degrees. And then, we remove the vertexes with minimum degree, and in fact, the removed vertexes have the core number as their degree. Then, we decrease the core number of the neighbors of the removed vertexes by one, which is the effect of removing the decreased vertexes's neighbors. Then we restart the remove step again and decrease again and loop. It is the basic idea, since we keep removing the vertex that can be removed for all the $\xi(v)$-cores that v is not to be removed now since they belong to a bigger or the same size core and this core will not be affected by this removal. And in fact, we remove only one vertex at one time to ensure the decremental is one by one and is correct.

Statistical Relaxation of Clique : η -Dense Subgraph

Clique is 1-dense subgraph for the reason that all the possible edge is real.

Density has a relation with centrality.

Definitions

density : $\varrho(G) = \frac{|E|}{|V|(|V|-1)}$

average degree : $\bar{d}\,(G) = \frac{|E|}{|V|}$

η-density subgraph : $U \subseteq V, s.t. \varrho(G[U]) \geq \eta$

N-density subgraph : $U \subseteq V, s.t. \bar{d}\,(G[U]) \geq \eta$

the maximum average degree : the maximum average degree of any non-empty induced subgraph of G, $\gamma^*(G) = max\{\bar{d}\,(U)|U \subseteq V \; and \; U \neq \emptyset\}$.

densest subgraph : $G([U]), s.t. \bar{d}\,(G[U]) = \gamma^*(G)$.

densest subgraph of size k : $G([U]), s.t. |U| = k \; and \; \bar{d}\,(G[U]) = \gamma^*(G, k)$.

walk of length ℓ : the number of vertexes a vertex can reach with just length ℓ.

degree of vertex v of order ℓ: a generalization of degree, denoted by $d_G^\ell(v)$, equals the walks of length ℓ started at *v*.

walk of length ℓ of G : sum of $d_G^\ell(v)$ over V, denoted by $W_\ell(G)$

density of order ℓ: $\varrho_\ell(G) = \frac{W_\ell(G)}{|V|(|V|-1)}$.

The notation is confused when we meet a phrase like "0.5-density subgraph", we don't know whether the 0.5 is just density and it is average degree.

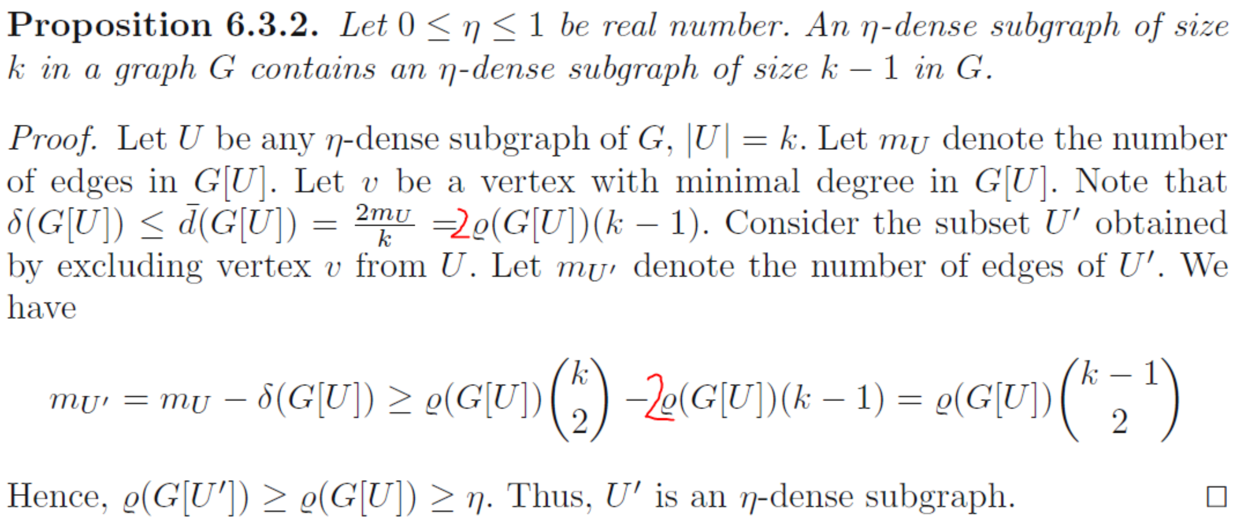
The relation between density and average degree is simple :

$(|U| - 1)\varrho(G[U]) = \bar{d}\,(G[U])$.

It is just a little bit like the relation between betweenness centrality and stress centrality. And η-density subgraph and N-density subgraph are both not closed under exclusion but the former is nested while the later is not.

Propositions

1. η-dense subgraph is nested.



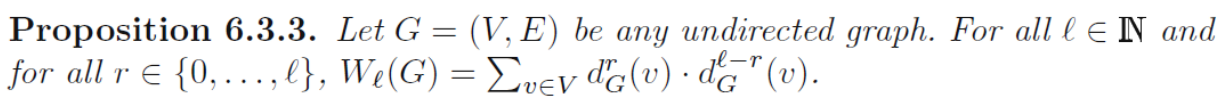
Proposition 6.3.2. *Let $0 \le \eta \le 1$ be real number. An η-dense subgraph of size k in a graph G contains an η-dense subgraph of size $k-1$ in G.*

Proof. Let U be any η-dense subgraph of G, $|U| = k$. Let m_U denote the number of edges in $G[U]$. Let v be a vertex with minimal degree in $G[U]$. Note that $\delta(G[U]) \le \bar{d}(G[U]) = \frac{2m_U}{k} = 2\varrho(G[U])(k-1)$. Consider the subset U' obtained by excluding vertex v from U. Let $m_{U'}$ denote the number of edges of U'. We have

$$m_{U'} = m_U - \delta(G[U]) \ge \varrho(G[U])\binom{k}{2} - 2\varrho(G[U])(k-1) = \varrho(G[U])\binom{k-1}{2}$$

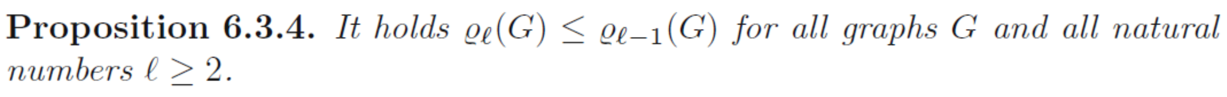
Hence, $\varrho(G[U']) \ge \varrho(G[U]) \ge \eta$. Thus, U' is an η-dense subgraph. □

2. to calculate the density of order ℓ.



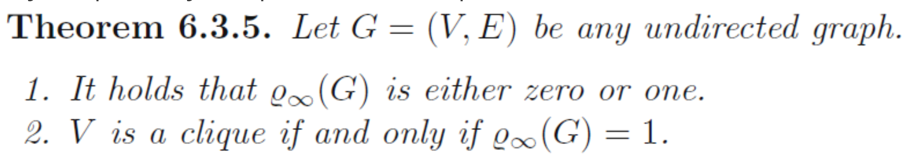
Proposition 6.3.3. *Let $G = (V, E)$ be any undirected graph. For all $\ell \in \mathbb{N}$ and for all $r \in \{0, \dots, \ell\}$, $W_\ell(G) = \sum_{v \in V} d_G^r(v) \cdot d_G^{\ell - r}(v)$.*

3. the bigger the order ℓ, the smaller the density.



Proposition 6.3.4. *It holds $\varrho_\ell(G) \le \varrho_{\ell-1}(G)$ for all graphs G and all natural numbers $\ell \ge 2$.*

4. in fact, we can scale our compactness measurement by increasing or decreasing ℓ, and if $\ell = \infty$ then only the perfectly compact one, the cliques, could have the value 1.



Theorem 6.3.5. *Let $G = (V, E)$ be any undirected graph.*

1. It holds that $\varrho_\infty(G)$ is either zero or one.
2. V is a clique if and only if $\varrho_\infty(G) = 1$.

Find $\gamma^*(G)$

There is a very clever algorithm to do this job by translating this question into a maximum flow problem. Which make use of the following facts:

1. every cut is smaller or equals than the cut $c(S, \bar{S})$, then there is an inequality we can use.
2. the number for the possible $\gamma^*(G)$ is limited.
3. with inequality and limited possible solution, a binary search can be performed.

So, we can find the $\gamma^*(G)$ in polynomial running time. The textbook has a clear description needs no explanation:

More specifically, the network is defined as

$$V' =_{\text{def}} V \cup \{s, t\}$$
$$E' =_{\text{def}} \{(v, w) \mid \{v, w\} \in E\} \cup \{(s, v) \mid v \in V\} \cup \{(v, t) \mid v \in V\}$$

and for $v, w \in V'$ the capacity function u_γ is defined as

$$u_\gamma(v, w) =_{\text{def}} \begin{cases} 1 & \text{if } \{v, w\} \in E \\ m & \text{if } v = s \\ m + \gamma - d_G(v) & \text{if } w = t \\ 0 & \text{if } (v, w) \notin E' \end{cases}$$

We consider capacities of cuts in the network. Let S, T be any partitioning of V' into two disjoint vertex sets with $s \in S$ and $t \in T$, $S_+ = S - \{s\}$ and $T_+ = T - \{t\}$. Note that $S_+ \cup T_+ = V$. If $S_+ = \emptyset$, then the capacity of the cut is $c(S, \overline{S}) = m|V|$. Otherwise we obtain:

$$\begin{aligned} c(S, T) &= \sum_{v \in S, w \in T} u_\gamma(v, w) \\ &= \sum_{w \in T_+} u_\gamma(s, w) + \sum_{v \in S_+} u_\gamma(v, t) + \sum_{v \in S_+, w \in T_+} u_\gamma(v, w) \\ &= m|T_+| + \left(m|S_+| + \gamma|S_+| - \sum_{v \in S_+} d_G(v) \right) + \sum_{\substack{v \in S_+, w \in T_+ \\ \{v, w\} \in E}} 1 \\ &= m|V| + |S_+| \left(\gamma - \frac{1}{|S_+|} \left(\sum_{v \in S_+} d_G(v) - \sum_{\substack{v \in S_+, w \in T_+ \\ \{v, w\} \in E}} 1 \right) \right) \\ &= m|V| + |S_+|(\gamma - \bar{d}(G[S_+])) \end{aligned} \tag{6.2}$$

It is clear from this equation that γ is our guess on the maximum average degree of G. We need to know how we can detect whether γ is too big or too small.

Then, the cardinality of S_+ matters. Because c(S, T) <= m|V|, so $|S_+| == 0$ only and only if $\gamma \geq \gamma^*(G)$. Then it would be the judgment of whether γ is too big.

Find the subset U of size k that $\bar{d}(U) = \gamma^*(G, k)$

It is NP-hard. A greedy algorithm to approximate the U with a approximation ratio:

Algorithm 12: Greedy procedure

```
Input: Graph $G = (V, E)$ and even parameter $k \in \mathbb{N}$ (with $|V| \geq k$)
Output: A set of $k$ vertices of $G$

Sort the vertices in decreasing order of their degrees;
Let $H$ be the set of $\frac{k}{2}$ vertices of highest degree;
Compute $N_H(v) = |N(v) \cap H|$ for all vertices $v \in V - H$;
Sort the vertices in $V - H$ in decreasing order of the $N_H$-values;
Let $R$ be the set of $\frac{k}{2}$ vertices of $V - H$ of highest $N_H$-values;
Return $H \cup R$
```

Theorem 6.3.8. *Let G be any graph with n vertices and let $k \in \mathbb{N}$ be an even natural number with $k \leq n$. Let $A(G, k)$ denote the average degree of the subgraph of G induced by the vertex set that is the output of Algorithm 12. We have*

$$\gamma^*(G, k) \leq \frac{2n}{k} \cdot A(G, k).$$

As we can see, there is three parameters: G, k, γ, and we have tried to get γ when give G, k. What if we are given $G, \gamma \; and \; k$ and be asked for the existence of such a N-dense subgraph? well, we know that is $\gamma = k - 1$ we can solve it by searching for clique of size k which is NP-complete, but if we have $\gamma = 0$, then it is no even a problem. So, how does the complexity increase with γ ? There is a theorem:

Theorem 6.3.10. *Let γ be any density threshold.*

1. *If $\gamma = 2 + \mathcal{O}\left(\frac{1}{k}\right)$, then γ-Dense Subgraph is solvable in polynomial time.*
2. *If $\gamma = 2 + \Omega\left(\frac{1}{k^{1-\varepsilon}}\right)$ for some $\varepsilon > 0$, then γ-Dense Subgraph is $\mathcal{NP}$-complete.*