The K-clique Percolation Method
Clique Clustering

Phil

May 1, 2023

Outline

Percolation

The Phase Transition™

Cliques

Clique Percolation Algorithm

Addenda

Σ

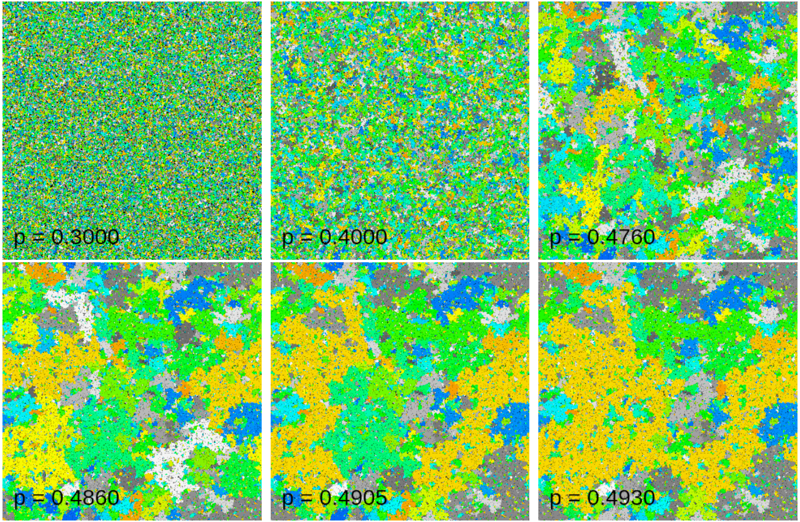
Section 1

Percolation

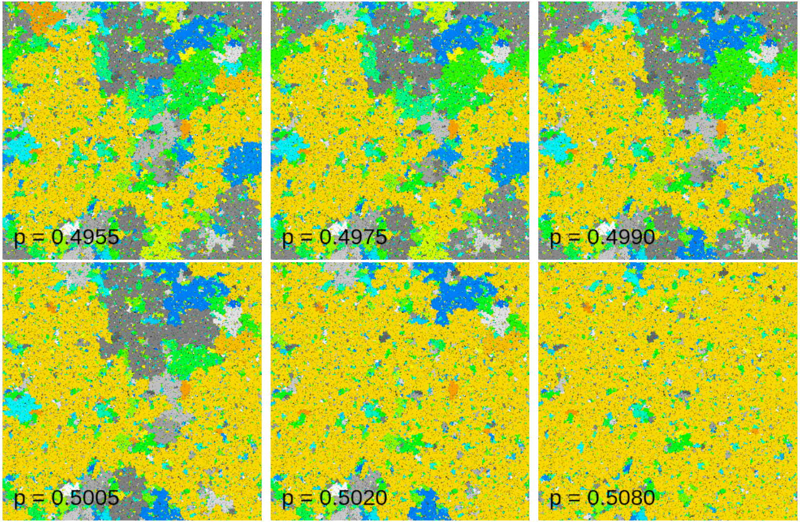
Percolating Water Problem

- Assume that some liquid is poured on top of some porous material. Will the liquid be able to make its way from hole to hole and reach the bottom?
  - Model: three-dimensional network of $n \times n \times n$ vertices, usually called "sites", in which the edge or "bonds" between each two neighbors may be open or closed
  - Probabilistic: A bond is open with probability p and closed $1 - p$.
- For a given p, what is the probability that an open path exists from the top to the bottom?

Below 50% Openness



Above 50% Openness



Section 2

The Phase Transition™

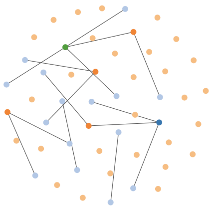
New Topic: Random Graphs [FK23]

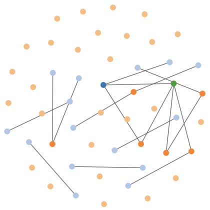
- Idea: Define some parameters and generate a graph probabilistically
- Erdős–Rényi Model: The most common model
- $G(n, p)$: Add n nodes, then add edges with probability p
- $G(n, M)$: Add n nodes, then pick uniformly from all configurations of M edges

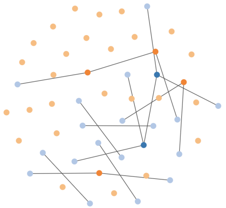
The "Percolated" Graph

- The ER graph's definition is closely related to the percolation problem defined earlier
- **G(n,p):** each edge has a fixed probability of being present or absent, independently of the other edges
- **An ER graph also has a percolation constant!**
- The evolution of the ER graph structure with increasing p can be very precisely proven [not in this presentation] but it is dominated by the same critical percolation constant of $p = \frac{1}{n}$

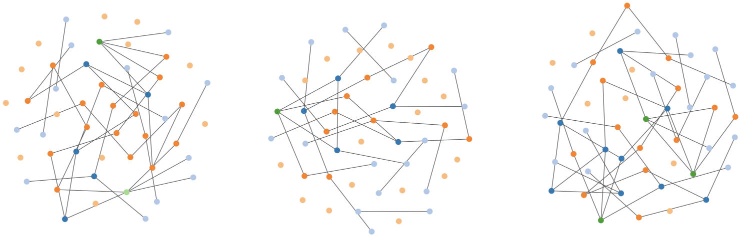
Erdős–Rényi Phase Transition: $p = 75\%\frac{1}{n}$







Erdős–Rényi Phase Transition $p = 125\%\frac{1}{n}$



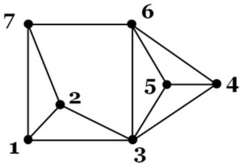
Generalizing

- The giant connected component is made up of nodes connected to *each other*
- The giant connected component is made up of pairs of nodes connected to **other, possibly overlapping pairs of nodes**
- Can we find denser "clusters" by finding **cliques** connected to other cliques? **Yes!**

Section 3

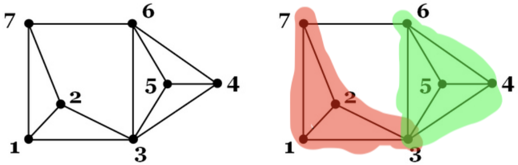
Cliques

Cliques Review



- Some definitions for review:
- *Definition:* A clique c is a fully connected set of nodes, i.e. every pair of its vertices is connected by a link in the graph.
- *Definition:* A k-clique c_k is a clique of k nodes.
- *Definition:* Two k-cliques are said to be adjacent if and only if they share $k-1$ nodes.

CPM Community



- *Definition:* A "clique percolation method" community is defined as the maximal union of k-cliques that can be reached from each other through a series of adjacent k-cliques
- How do we intuit what this means?
- Start with a clique "template", and "roll it over" on the graph. {1, 2, 7} rolls over to {1, 2, 3}
- Alternatively, the "Group Chat of Theseus": take one node out of the clique, include a new one.
- Where are the 3-clique communities on this graph?

Questions?

Section 4

Clique Percolation Algorithm

Motivation

- We want to understand the structure of a graph - the "communities" within it
  - ex. social media - what topics or communities someone participates in or follows
  - ex. biology - identify related proteins within an interaction network
- Our approach (overlapping cliques) is an intuitive and deterministic definition of a community
- Allows for overlapping communities - many other methods don't do this
- Density requirement is freely adjustable via k

Algorithm Requirements

- *Subtask:* find the cliques
  - Either find all maximal cliques, or find all k-cliques in a graph
  - These are both NP-hard problems! Outside the scope of this lecture
  - A recent CPM paper uses a parallelized algorithm [Dan18]
- *Subtask:* storing the communities / connected components
  - We will use a **Union-Find data structure** (disjoint sets)
  - UF.MakeSet(): creates a new tree with one node p, corresponding to a new empty set, and returns p.
  - UF.Find(p) returns the root of the tree
  - UF.Union($r_1, ..., r_l$): performs the union of trees represented by their roots r_i by making one root the parent of all others.

Pseudocode

BASIC CPM ALGORITHM(G):

UF $\leftarrow$ Union-Find data structure
DICT $\leftarrow$ []
for each k-clique $c_k \in G$ **do**:
    $S \leftarrow \{\}$
    for each $(k-1)$-clique $c_{k-1} \subset c_k$ **do**:
        if $c_{k-1} \in$ DICT.KEYS() **then**
            $p \leftarrow$ UF.FIND(DICT[c_{k-1}])
        else
            $p \leftarrow$ UF.MAKESET()
            DICT[c_{k-1}] $\leftarrow$p
        $S \leftarrow S \cup \{p\}$
    UF.UNION(S)

Questions?

Section 5

Addenda

Percolation Threshold

- There are also threshold thresholds for clique communities. What edge probability p is the threshold to producing one giant connected community in an ER graph?
- Intuition: going back to the "rolling-over" analogy. Within a "rolling", at the threshold, there should be, in expectation, one adjacent clique at each clique, to "continue" to build the cluster.
- $(k-1) \times (N-k-1)p_c^{k-1} = 1$: candidate vertices to remove $\times$ candidate vertices to add
- For large N, this comes out to $N(k-1)p_c^{k-1} = 1$ [Der05]

$$p_c(k) = \frac{1}{[N(k-1)]^{\frac{1}{k-1}}} \tag{1}$$

Optimizations

- The given CPM algorithm is prohibitively expensive in memory because we store large number of $(k-1)$-cliques
- *Optimization:* instead of disjoint sets of $(k-1)$ cliques, consider non-disjoint sets of z-cliques s.t. $z < (k-1)$ [Bau22]
- why are there fewer z-cliques? Consider binomial theorem, $k \ll M$
- this leads to an *approximation* of the previous algorithm, but it is otherwise very similar and produces very similar results

Questions?

Bibliography

A. Baudin.
Clique percolation method: Memory efficient almost exact communities.
Feb 2022.

Balalau O. Sozio M. Danisch, M.
Listing k-cliques in sparse real-world graphs.
2018.

Palla G. Vicsek T. Derényi, I.
Clique percolation in random networks.
Physical Review Letters, 94(46), 2005.

A. Frieze and M. Karonski.
Introduction to Random Graphs.
2023.