University of Twente

Master's Thesis Applied Mathematics

Algorithms for planted clique recovery in random geometric graphs

Siemo Zhang

Supervisor
Prof.dr. N.V. Litvak
n.litvak@utwente.nl

Graduation Committee
Prof.dr. N.V. Litvak
Dr. B. Manthey
Dr. A.V. Bobu

UNIVERSITY
OF TWENTE.

Preface

For the past seven months I have been working on my graduation project to conclude my Master's programme Applied Mathematics. Shortly after I started, the COVID-19 virus spread to Europe and I was forced to work from home. This was definitely suboptimal and it was hard to always find the motivation. Luckily I got over this hurdle with the help and support of my friends and family, so that I can gladly present this thesis to you now.

I want to thank Nelly for all the help she provided me. Even though we could not meet in person due to the pandemic, our online meetings were always helpful, an guided me towards the right direction for my thesis. I am also very grateful for the detailed feedback that was given for my thesis draft, not only could I to polish my thesis more, but I also learned a thing or two on the techniques of writing a readable and cohesive piece of text.

Furthermore I want to thank Andrei and Konstantin for introducing the topic to me during my internship, the previously acquired knowledge helped me immensenly in the process of writing this thesis.

Lastly I want to thank both Andrei and Bodo for joining the assessment committee and taking the time of reading/evaluating my thesis.

Abstract

Let $G(n, r)$ be a random geometric graph. A random geometric graph is an undirected graph that has edges between vertices if and only if the distance between vertices does not exceed r, the neighbourhood radius. We assume the number of points in the random geometric graph to be n and the points are uniformly placed in a two dimension unit flat square torus.
We select k points out of n points from the random geometric graph. Let the set of k points be denoted by K. We add edges between each pair of vertices in K, such that K becomes a planted clique. In this thesis we investigate the problem of recovering the planted clique K in a random geometric graph $G(n, r)$ for large n. We introduce two methods to almost surely recover the planted clique. One method is based on counting the number of common neighbours between vertex pairs, and is able to solve the problem for a large range of k. Another method is based on the vertex degree of each vertex in the graph, it is able to solve the problem when k is equal to the average degree of vertices in the graph.

Contents

1 Introduction

A random geometric graph $G(n, r)$ is a graph of n points placed randomly in a metric space, where two vertices are adjacent if and only if their euclidean distance is less than r.

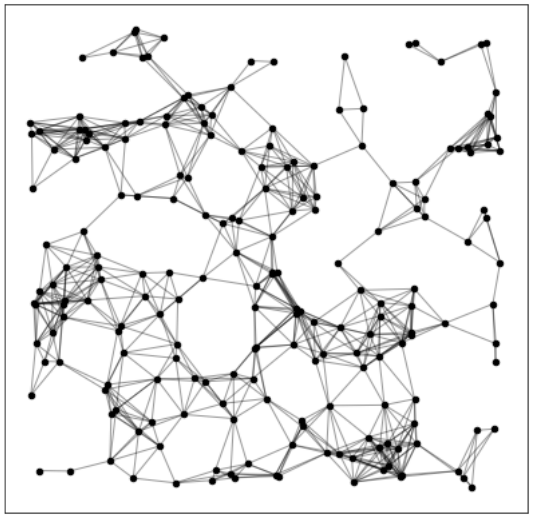


Figure 1: Random geometric graph $G(n, r)$ with $n = 200$ and $r = 0.125$

Random Geometric Graphs are related to real world scenarios in many ways. An important property of random geometric graphs is the community structure, vertices in the graph can be grouped into sets of vertices, such that each set of vertices is internally densely connected. This is a common phenomenon in networks of the real world; neighbours that live next to each other are more likely to be friends than a group of strangers, lions in a close group are more likely to be in the same pride than completely random lions[7]. Another property of the random geometric graph is that the vertices display degree assortativity: Vertices with high degree are likely connected to other vertices with high degree; In the social networking site Twitter, a person with high number of followers will more likely follow another person with high amount of followers[8].

There is very little literature on the topic of recovering a planted clique in random geometric graphs, a similar topic that has been studied much more is the recovery of planted cliques in Erdős–Rényi graphs. Kučera[5] (1995) observed that in the case of Erdős–Rényi graphs, as long as the planted clique is large enough ($k > c\sqrt{n \log(n)}$), then with high probability a vertex in the planted clique has higher degree than a vertex outside of the planted clique. Another algorithm exists [4] for the recovery of planted clique in Erdős–Rényi graphs, it is an algorithm based on the spectral properties of the graph by investigating the second eigenvector of the adjacency matrix. Unlike the algorithm based on vertex degrees however, the proof of this algorithm leverage specific properties of the Erdős–Rényi graph and cannot be replicated for the recovery of planted cliques in Random Geometric graphs.

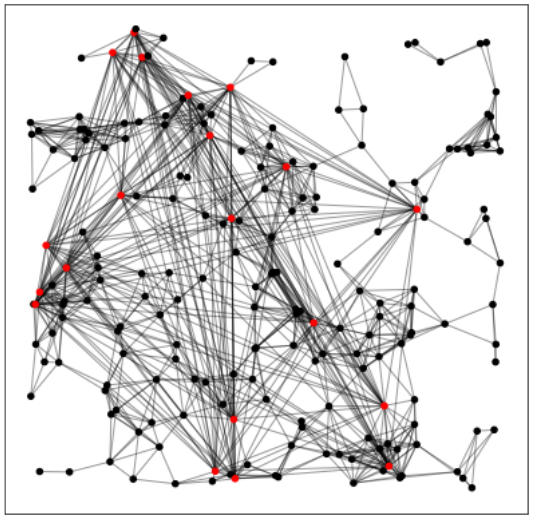


Figure 2: Random geometric graph $G(n, r)$ with $n = 200$, $r = 0.125$ and $k = 20$. Planted clique is displayed in red

In this thesis a random geometric graph $G(n, r)$ is placed onto a two dimensional unit flat square torus. We consider the case of very large n, where the planted clique is of size k. The goal of this thesis is to find optimal algorithms in terms of complexity for solving the problem in different regimes of the size k of the planted clique and neighbourhood radius r, where the algorithms are required to recover the planted clique with high probability (w. h. p.). The detection of the planted clique varies in difficulty depending on the size of the clique k and the value of r_n where $\lim_{n\to\infty} r_n \to 0$. In general it is easier to find the planted clique for large k and small r compared to large r and small k, because then the vertices in the planted clique stand out with their high vertex degree compared to regular vertices.

In this thesis we present two algorithms for the recovery of planted cliques in random geometric graphs. We first present the general intuition of the algorithm as there exists no previously similar work. We then proof the correctness of the algorithms in their respective areas of applicability. We show the theoretical complexity of the algorithms. Furthermore, we conduct practical experiments by implementing the algorithms in Python, applying the algorithms to randomly generated random geometric graphs with planted clique. We show that the algorithms do not only work in theory, but also have good performance in practice.

In Chapter 1.1 we formally introduce the problem of recovering a planted clique in random geometric graphs. Any definitions that will be used onwards are explained in this chapter. In Chapter 2 we introduce the neighbourhood algorithm, a method that depends on counting the number of common neighbours between pairs of vertices to recover the planted clique. This method is most notable for recovering planted clique of size $O(1)$. In Chapter 3 we introduce the Vertex Removal algorithm, a method that recovers the planted clique by iterating the removal of low degree vertices. This method is most notable for planted clique recovery of average vertex degree size $(\pi n r^2)$. In Chapter 4 we run practical experiments on the algorithms. We study the accuracy of the algorithm with varying values of n (graph size), r (neighbourhood radius) and k (planted clique size), to see if the theoretical area of applicability applies in practice. We then study the practical run-time of

each algorithm to get an idea on the average complexity.

1.1 Terminology

Let $G = (V, E)$ define a graph with:

1. A set of vertices V, where $|V| = n$.
2. A set of edges E, where $|E| = m$.

The following definitions must be introduced before we further explore the problem of planted clique recovery in random geometric graphs.

Definition 1. *A **random geometric graph (RGG)**, $G(n, r)$ is an undirected graph with n vertices distributed randomly on a metric space $[0, 1)^d$. Two vertices $i, j \in V$ are connected if and only if their euclidean distance is less than the connectivity radius $r \in (0, 1)$. A random geometric graph with planted clique of size k is denoted by $G(n, r, k)$*

Definition 2. *A **clique** is a (sub)graph of which each pair of vertices are connected by an edge.*

Definition 3. *A **planted clique** is a clique formed by selecting a subset of vertices in a graph, and adding edges between each pair of vertices in the subset.*

Definition 4. *The **clique number** of G is the number of vertices in the maximum clique of G.*

The clique number of $G(n, r)$ is then denoted by $\omega(n, r)$, which depends on the chosen values of n, r and can be separated in the following regimes [2].

1. $nr^2 \leq n^{-a}$ for some fixed $a > 0$, then: $lim_{n\to\infty} P(\omega(n, r) = 2) = 1$

2. $n^{-\epsilon} \ll nr^2 \ll \log(n)$
   for all $\epsilon > 0$, then:
   $\omega(n, r) = \Theta\left(\frac{\log(n)}{\log\left(\frac{\log(n)}{nr^2}\right)}\right) \leq \log(n)$

3. $nr^2 \gg \log(n)$, then: $\omega(n, r) = \Theta(nr^2)$

With this, the research problem can be fully understood. An important property of the vertices in a random geometric graph is the average degree. It is hard to recover a planted clique of average degree size, since the planted clique vertices will then have a similar vertex degree as vertices outside the planted clique. It is easy to find that vertices $v \in V$ in a random geometric graph $G(n, r)$ have an average vertex degree of πnr^2. Due to the properties of the random geometric graph $G(n, r)$, a vertex only has connections to vertices within a radius r of itself as shown in figure 3. The neighbourhood area of a vertex $v \in V$ is a circle of size πr^2.

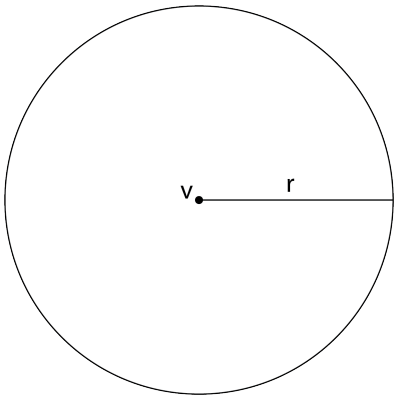


Figure 3: Neighbourhood area of vertex v

Thus the vertex degree of any vertex $v \in V$ is distributed by $D_v \sim \text{Poisson}(\pi n r^2)$. Leading to an average vertex degree of $\mathbb{E}(D_v) = \pi n r^2$
Let m denote the edge count of the graph $G(n, r)$. The edge count m is often used in describing the computational complexity of the algorithms, since oftentimes the computational complexity of an algorithm that solves the problem planted clique recovery in any graph is largely influenced by the number of edges in the graph. To get a better idea of what order of magnitude m is, it is useful to to express m in terms of n and r. The following expression for m always holds, where $f(n)$ is any monotone increasing function of n:

Theorem 1. *Let $G(n, r)$ be a random geometric graph, and let m be the number of edges in this graph. Then for large n, we have that*

$$\lim_{n\to\infty} P\big(m \leq n^2 r^2 + nrf(n)\big) = 1. \tag{1}$$

Proof. Let $p = \pi r^2$ denote the probability that an edge exists between two arbitrary vertices in the graph. Let $Y_i j$ be the random variable that is equal to 1 if an edge between pair i exists and zero otherwise, then $\mathbb{E}[Y_i] = p$. The edge count m is a sum of $\binom{n}{2}$ random variables Y_i, or $m \sum_{i=1}^{n(n-1)/2} Y_i$,

such that m has the following expected value and variance:

$$\mathbb{E}[m] = \mathbb{E}[\sum_{i=1}^{n(n-1)/2} Y_i] \tag{2}$$

$$= \sum_{i=1}^{n(n-1)/2} \mathbb{E}[Y_i] \tag{3}$$

$$= \sum_{i=1}^{n(n-1)/2} p = \binom{n}{2}\pi r^2, \tag{4}$$

$$\mathrm{Var}[m] = \mathrm{Var}[\sum_{i=1}^{n(n-1)/2} Y_i] \tag{5}$$

$$= \mathbb{E}[(\sum_{i=1}^{n(n-1)/2} Y_i)^2] - (E[\sum_{i=1}^{n(n-1)/2} Y_i)])^2 \tag{6}$$

$$= \mathbb{E}[\sum_{i=1}^{n(n-1)/2} (Y_i)^2 + 2\sum_{i\neq j} Y_i Y_j] - (E[\sum_{i=1}^{n(n-1)/2} Y_i)])^2 \tag{7}$$

$$= \sum_{i=1}^{n(n-1)/2} \mathbb{E}[(Y_i)^2] + 2\sum_{i\neq j} \mathbb{E}[Y_i Y_j] - \sum_{i=1}^{n(n-1)/2} (E[Y_i])^2 - 2\sum_{i\neq j} \mathbb{E}[Y_i]\mathbb{E}[Y_j] \tag{8}$$

$$= \sum_{i=1}^{n(n-1)/2} \mathbb{E}[(Y_i)^2] + 2\sum_{i\neq j} \mathbb{E}[Y_i]\mathbb{E}[Y_j] - \sum_{i=1}^{n(n-1)/2} (E[Y_i])^2 - 2\sum_{i\neq j} \mathbb{E}[Y_i]\mathbb{E}[Y_j] \tag{9}$$

$$= \sum_{i=1}^{n(n-1)/2} \mathbb{E}[(Y_i)^2] - \sum_{i=1}^{n(n-1)/2} (E[Y_i])^2 \tag{10}$$

$$= \sum_{i=1}^{n(n-1)/2} \mathrm{Var}[Y_i] \tag{11}$$

$$= \frac{n(n-1)}{2} p(1-p) = \frac{n(n-1)}{2}\pi r^2(1-\pi r^2), \tag{12}$$

$$\mathbb{P}(|m - \mathbb{E}[m]| \geq t) \leq \frac{\mathrm{Var}[m]}{t^2} \tag{13}$$

$$= \frac{\frac{n(n-1)}{2}\pi r^2(1-\pi r^2)}{t^2} \tag{14}$$

$$\leq \frac{\frac{n(n-1)}{2}\pi r^2}{t^2}. \tag{15}$$

We have that $\mathbb{P}(|m - \mathbb{E}[m]| \geq t) \to 0$ as $n \to 0$ if $t = nrf(n)$, and thus:

$$\lim_{n\to\infty} P(m \leq \mathbb{E}[m] + nrf(n)) = 1, \tag{16}$$

$$\lim_{n\to\infty} P(m \leq \frac{n(n-1)}{2}\pi r^2 + nrf(n)) = 1, \tag{17}$$

$$m = O_p(n^2r^2 + nrf(n)). \tag{18}$$

□

Note that m is not necessarily binomially distributed, because the event that both edges (i, j) and (i, l) exist, and the event that the edge (j, l) exists are dependent. but the proof makes use of the fact that in random geometric graphs every two edges are pair wise independent from each other. In this thesis we assume a lower bound of $r = \Omega\left(\frac{1}{\sqrt{n}}\right)$ for the connectivity radius, since any lower value of r causes the graph to contain no edges by Theorem 1. Recovering a planted clique by using only the adjacency matrix should then no longer be difficult.

1.2 Notations

The notation 'w. h. p.' will be used often in this thesis due to its simplicity, and has the following meaning.

Definition 5. *An event X_n whose probability depends on number n happens with high probability (w. h. p.) if $lim_{n\to\infty} P(X) = 1$*

Another notation that will often be used is "$\gg$" and "$\ll$" for the sake of simplicity as well, since denoting limits every time makes the derivations harder to read.

Definition 6. *For functions $f(n)$ and $g(n)$, if $f(n) \gg g(n)$ or $g(n) \ll f(n)$ then*

$$\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0 \tag{19}$$

Furthermore we introduce a list of symbols that will be used often in the next sections.

Symbol	Usage
$G(n, r)$	Random geometric graph of size n with neighbourhood radius r
$G(n, r, k)$	Random geometric graph of size n with neighbourhood radius r and planted clique of size k
n	The number of vertices in the graph
r	In a random geometric graph, the neighbourhood radius
k	The size of the planted clique
K	The vertices of the planted clique
D_i	Degree distribution of vertex i
$N(v)$	Set of vertices in the neighbourhood of vertex v
$N_{(i,j)}$	The number of common neighbours between vertices i and j
$f(n), g(n)$	any monotone increasing function in n, unless stated otherwise
$\omega(n, r)$	The clique number of random geometric graph $G(n, r)$
$d(i, j)$	The euclidean distance between vertices i and j.

2 Neighbourhood Algorithm

2.1 Method

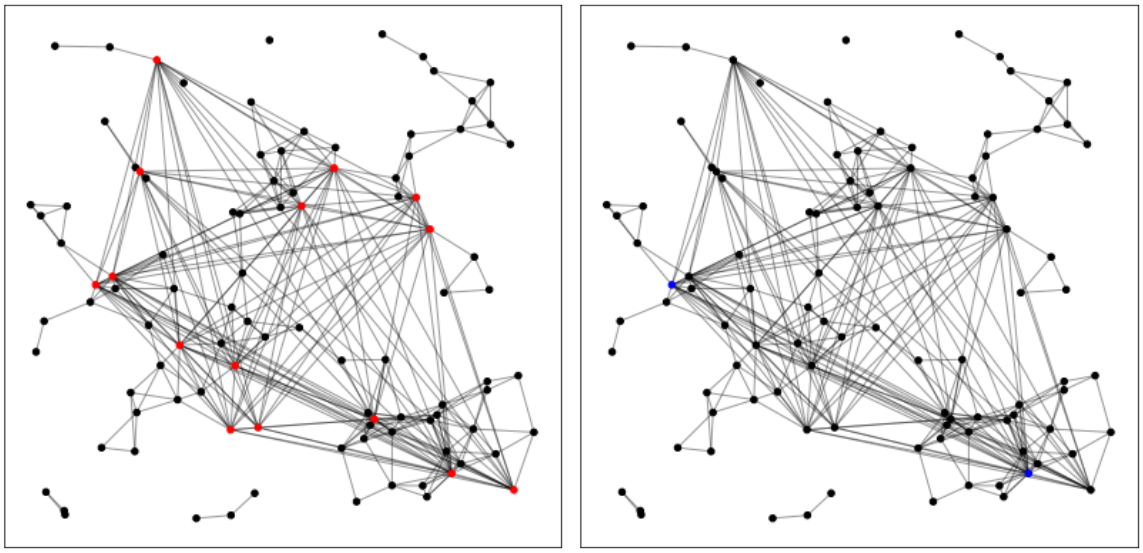
The random geometric graph has interesting properties. In the case of a random geometric graph placed onto a unit flat square torus, the euclidean distance between any pair of vertices $u, v \in V$ directly correlates to the number of common neighbours between vertices u and v; the larger the distance, the less common neighbours present. The common neighbourhood area between vertices u, v decreases as the euclidean distance $d(u, v)$ is increased, reaching a value of 0 when $d(u, v) > 2r$. When the common neighbourhood area is 0 between two vertices u, v, the vertices have no common neighbours at all. We know that planted clique vertices are typically far away from each other (distance $> 2r$) since the vertices were sampled randomly, thus giving us the idea of counting the number of common neighbours of the endpoints of each edge in the graph to find the planted clique.

Suppose the size of the planted clique k differs sufficiently from the average vertex degree of a graph $G(n, r)$, then by intuition the vertices of a randomly selected edge from $G(n, r)$ will not have $k - 2$ common neighbours. The vertices of an edge taken from the planted clique in $G(n, r, k)$ will have at least $k - 2$ common neighbours. Most of the time the vertices of an edge from the planted clique will in fact have exactly $k-2$ common neighbours, since a random vertex pair from the planted clique will likely exceed distance $2r$, it means that the vertex pair will likely have no common neighbours in the original graph $G(n, r)$. Thus aside from other planted clique vertices the vertex pair will not have many additional common neighbours. We check adjacent vertex pairs until a pair is found with exactly $k - 2$ common neighbours, we claim that the vertex pair and its common neighbours is our planted clique. The Neighbourhood Algorithm is also described using the following pseudo code.

```
Neighbourhood algorithm

input: G(n,r,k), G=(V,E)
output: planted clique K
K=[]
for all edges in E:
    find common neighbours of the end points
    if number of common neighbours=k-2:
        add edge to K
        add common neighbours to K
        break
return K
```

To demonstrate how this algorithm functions, first look at the initial graph $G(n, r, k)$ where the planted clique is marked in red. Consider the situation where k **is sufficiently large** (with the exact bounds given in the next subsection).



(a) $G(n, r, k)$ with planted clique K marked in red. Where $n = 100$, $r = 0.125$ and $k = 15$.

(b) The vertex pair found by the Neighbourhood algorithm with exactly $k - 2$ common neighbours.

Figure 4: The neighbourhood algorithm.

By having a large planted clique, only a pair of vertices in the planted clique $i, j \in K$ can have exactly $k - 2$ common neighbours. The algorithm checks the endpoints for every single edge in the graph, until a pair of vertices is found with exactly $k - 2$ vertices as seen in figure 4b. By only displaying the vertex pair and its common neighbours, the resulting graph is the planted clique. (figure 5)

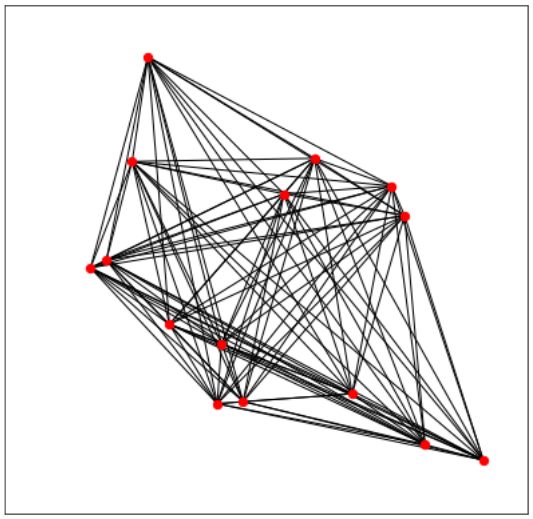


Figure 5: The planted clique recovered by the Neighbourhood algorithm

Not only can the Neighbourhood algorithm recover the planted clique for large k, but it is also able to recover the planted clique if k is significantly smaller than the average vertex degree, and if the graph is dense enough. Consider the case where k **is sufficiently small**. Here the planted clique is marked again in red in figure 6.

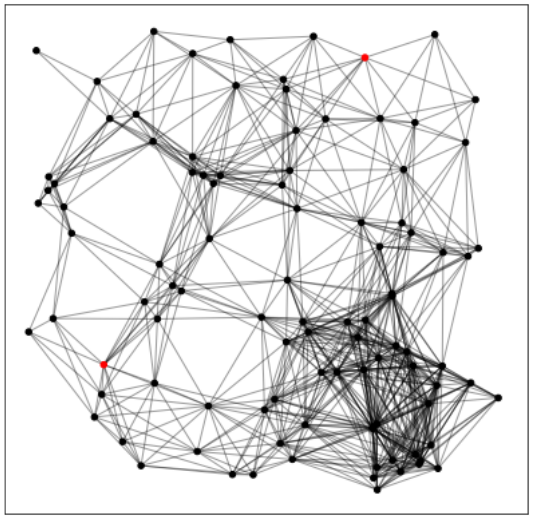


Figure 6: Dense graph with $n = 100$, $r = 0.25$ and small planted clique (planted edge) $k = 2$

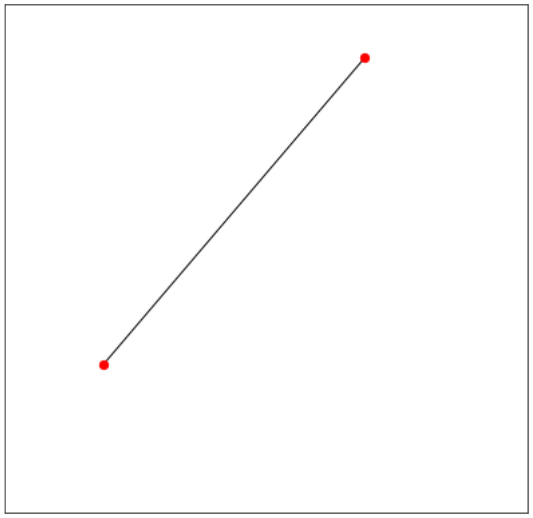


Figure 7: Small planted clique recovered by Neighbourhood algorithm

The planted clique here consists of two vertices ($k = 2$), and surprisingly enough the only vertex pair of an edge in this graph with no common neighbours is the pair of the planted clique vertices. This is due to the fact that it is rare for the endpoints of an edge to have a small number of common

neighbours if the graph is dense. Since the vertices in the planted clique are chosen randomly, a pair of vertices in the planted clique have a geometrical distance of $> 2r$ w.h.p. (with high probability). This means a randomly chosen pair of vertices in the planted clique likely has no common neighbours that do not belong to the planted clique. So for small k, any vertex pair from the planted clique will also have small number of common neighbours. The planted clique is then recovered by taking the the vertex pair and its common neighbours.

2.2 Area of applicability

The neighbourhood algorithm is interesting as it can recover the graph for different sizes of the planted clique. The following theorem gives us three statements: the first statement corresponds to the case where k is sufficiently large, the second and third statement correspond to the case where k is significantly smaller than average degree and graph is sufficiently dense.

Theorem 2. *Let $r \to 0$ as $n \to \infty$ and let one of the following set of conditions hold:*

1. $k \geq (1+\epsilon)\pi e n r^2$ *and* $k - 2 \geq 2/\ln(1+\epsilon)\ln(n)$ *for some* $\epsilon > 0$.
2. $1 \ll k \ll nr^2$ *and* $nr^2 \gg \frac{1}{2/3\pi - \sqrt{3}/2} \ln(\frac{n^2}{\sqrt{k-2}})$.
3. k *is fixed and* $nr^2 \geq \frac{2+\delta}{2/3\pi - \sqrt{3}/2} \ln(n)$ *for some* $\delta > 0$.

Then the neighbourhood algorithm recovers the planted clique with high probability.

Proof. The first and third statement are proved in an unfinished manuscript [9].

To prove the second statement of the theorem, one needs to show that with high probability no vertex pair of an edge outside of the planted clique has exactly $k-2$ common neighbours. Let us denote by $N_{(i,j)}$ the number of common neighbours of vertex pair i and j in random geometric graph $G(n,r)$ before planting the clique. And let $d(i,j)$ denote the euclidean distance between vertices i and j. Clearly,

$$P(N_{(i,j)} = k-2) \leq \sup_{0 \leq x \leq r} P(N_{(i,j)} = k-2 | d(i,j) = x). \tag{20}$$

This probability depends on the euclidean distance $d(i,j)$ between vertices i and j. By definition of the random geometric graph we have that $0 \leq d(i,j) \leq r$ if i,j are connected by an edge. A common neighbour of the vertices i and j should be located in the area of intersection of the neighbourhood areas of vertices i and j. Let $x = d(i,j)$, then the area of intersection can be described using the following expression:

$$A(r,x) = 2r^2 \arccos\left(\frac{x}{2r}\right) - \frac{1}{2} x \sqrt{4r^2 - x^2}. \tag{21}$$

The number of common neighbours $N_{(i,j)}$ is dependent on this area, and is Poisson distributed with mean $nA(r,x)$. The area $A(r,x)$ monotonously decreases from πr^2 to $(\frac{2}{3}\pi - \sqrt{3}/2)r^2$ as x increases from 0 to r. And since by assumption we have $k \ll nr^2$, we can express the following inequality:

$$\begin{aligned} k <& (\frac{2}{3}\pi - \sqrt{3}/2) n r^2 && (22)\\ =& \mathbb{E}[N_{(i,j)} | d(i,j) = r] && (23)\\ <& \mathbb{E}[N_{(i,j)} | d(i,j) = x] \quad \text{for } 0 \leq x < r. && (24) \end{aligned}$$

For k significantly smaller than the average degree (πnr^2) and a dense graph, the typical vertex in the graph $G(n,r)$ has significantly more neighbours than $k-1$. The same holds for the expected number of common neighbours $\mathbb{E}[N_{(i,j)}]$, it is always significantly larger than $k-2$ since the number of common neighbours $N_{(i,j)}$ is of the same order as average degree. This means that $\mathbb{E}[N_{(i,j)}]$ has a value closest to $k-2$ when the distance between vertices i,j is as large as possible, which is $d(i,j)=r$. The value of the expectation tells us that the probability $P(N_{(i,j)}=k-2|d(i,j)=x)$ reaches a supremum at $x=r$, given that the number of common neighbours $(N_{(i,j)}|d(i,j)=x)$ is Poisson distributed with a mean of $nA(r,x)$.

$$P(N_{(i,j)}=k-2) \leq \sup_{0\leq x\leq r} P(N_{(i,j)}=k-2|d(i,j)=x) \tag{25}$$

$$= P(N_{(i,j)}=k-2|d(i,j)=r) \tag{26}$$

$$\leq \frac{((2/3\pi-\sqrt{3}/2)nr^2)^{k-2}e^{-(2/3\pi-\sqrt{3}/2)nr^2}}{(k-2)!}. \tag{27}$$

By union bound,

$$P(\exists(i,j)\in E : N_{(i,j)}=k-2) \leq n^2\frac{((2/3\pi-\sqrt{3}/2)nr^2)^{k-2}e^{-(2/3\pi-\sqrt{3}/2)nr^2}}{(k-2)!}. \tag{28}$$

Let $c=2/3\pi-\sqrt{3}/2$ and using Stirling's Approximation,

$$P(\exists(i,j)\in E : N_{(i,j)}=k-2) \leq n^2\frac{(cnr^2)^{k-2}e^{k-2}e^{-cnr^2}}{\sqrt{k-2}(k-2)^{k-2}}. \tag{29}$$

$$\tag{30}$$

For $\lim_{n\to\infty} P(\exists(i,j)\in E : N_{(i,j)}=k-2)=0$ to be true we need the following to hold,

$$\lim_{n\to\infty} n^2\frac{(cnr^2)^{k-2}e^{k-2}e^{-cnr^2}}{\sqrt{k-2}(k-2)^{k-2}} = 0, \tag{31}$$

$$\iff \sqrt{k-2}(k-2)^{k-2}e^{cnr^2} \gg n^2(cnr^2)^{k-2}e^{k-2}, \tag{32}$$

$$e^{cnr^2} \gg \frac{n^2(cnr^2)^{k-2}e^{k-2}}{\sqrt{k-2}(k-2)^{k-2}}, \tag{33}$$

$$nr^2 \gg \frac{1}{c}\ln(\frac{n^2}{\sqrt{k-2}}) + \frac{1}{c}(k-2)\ln(\frac{cner^2}{k-2}). \tag{34}$$

$$\tag{35}$$

Let us now show that we can omit the second term of the inequality by using the given condition of

$k \ll nr^2$:

$$e^{\frac{nr^2}{k}} \gg \frac{nr^2}{k}, \quad \left(\frac{nr^2}{k} \gg 1\right), \tag{36}$$

$$e^{nr^2} \gg \left(\frac{nr^2}{k}\right)^k, \tag{37}$$

$$e^{nr^2} \gg e^{k \ln\left(\frac{nr^2}{k}\right)}, \tag{38}$$

$$nr^2 \gg k \ln\left(\frac{nr^2}{k}\right), \tag{39}$$

$$nr^2 \gg \frac{1}{c}(k-2)\ln(\frac{cner^2}{k-2}). \tag{40}$$

We conclude that in $G(n,r)$, with high probability, no two endpoints of an edge have exactly $k-2$ common neighbours, if the following holds:

$$nr^2 \gg \frac{1}{c}\ln(\frac{n^2}{\sqrt{k-2}}) \tag{41}$$

$$= \frac{1}{2/3\pi - \sqrt{3}/2}\ln(\frac{n^2}{\sqrt{k-2}}). \tag{42}$$

□

2.3 Complexity

Finding every edge has a complexity of $\mathcal{O}\left(n^2\right)$ or $\mathcal{O}\left(m\right)$. To find the common neighbours of the end points of each edge, we need to make n comparisons per edge. So the complexity of this action is n^3 or $\mathcal{O}\left(mn\right)$. Checking whether the common neighbours of each edge form a clique has a complexity of $\mathcal{O}\left(n^2k^2\right)$ or $\mathcal{O}\left(mk^2\right)$ if k is larger than the clique number $\omega(n,r)$, else the complexity of this checking procedure has a complexity of $\mathcal{O}\left(n^2\omega(n,r)\right)^2$ or $\mathcal{O}\left(m\omega(n,r)\right)$. The total complexity is then:

1. if $k > \omega(n,r)$: $\mathcal{O}\left(n^2k^2 + n^3\right)$ or $\mathcal{O}\left(mk^2 + mn\right)$.
2. if $k < \omega(n,r)$: $\mathcal{O}\left(n^2\omega(n,r)^2 + n^3\right)$ or $\mathcal{O}\left(mk^2 + mn\right)$.

A downside of this algorithm is the fact that checking every single pair of vertices is computationally inefficient, so a possibility of improvement to the run-time is sampling a subset $S \subset V$ of the vertices in the graph that has size much smaller than n but is still large enough to contain at least two vertices from the planted clique K with high probability. Then we check adjacent vertex pairs within this subset S until a pair is found with exactly $k-2$ common neighbours.

3 Vertex Removal algorithm

3.1 Method

In 1995 Kucera [5] observed that in Erdos Renyi graphs, for large enough k, the planted clique can be recovered by simply finding the k largest degree vertices. Although the specific threshold of k is different, the general idea that a planted clique can be recovered by finding the top k degree vertices is not only limited to Erdos Renyi graphs, but can also be applied to random geometric graphs with planted clique. Following the same train of thought, a planted clique in random geometric graphs can be recovered if all low degree vertices are removed.

Suppose we have a random geometric graph $G(n, r)$, and suppose that k is sufficiently large. By definition of a clique, we know that vertices in the planted clique $i \in K$ have $deg(i) \geq k - 1$. By removing vertices of degree $deg(v) < k - 1$ we can narrow down the search for the planted clique. After this procedure more vertices will appear with vertex degree smaller than $k - 1$. Repeat the vertex removal step until no more vertices can be removed. In this chapter we will identify conditions such that after the algorithms terminates, the only vertices left will be from the planted clique.

The Vertex Removal algorithm implementation is described by the following pseudo code,

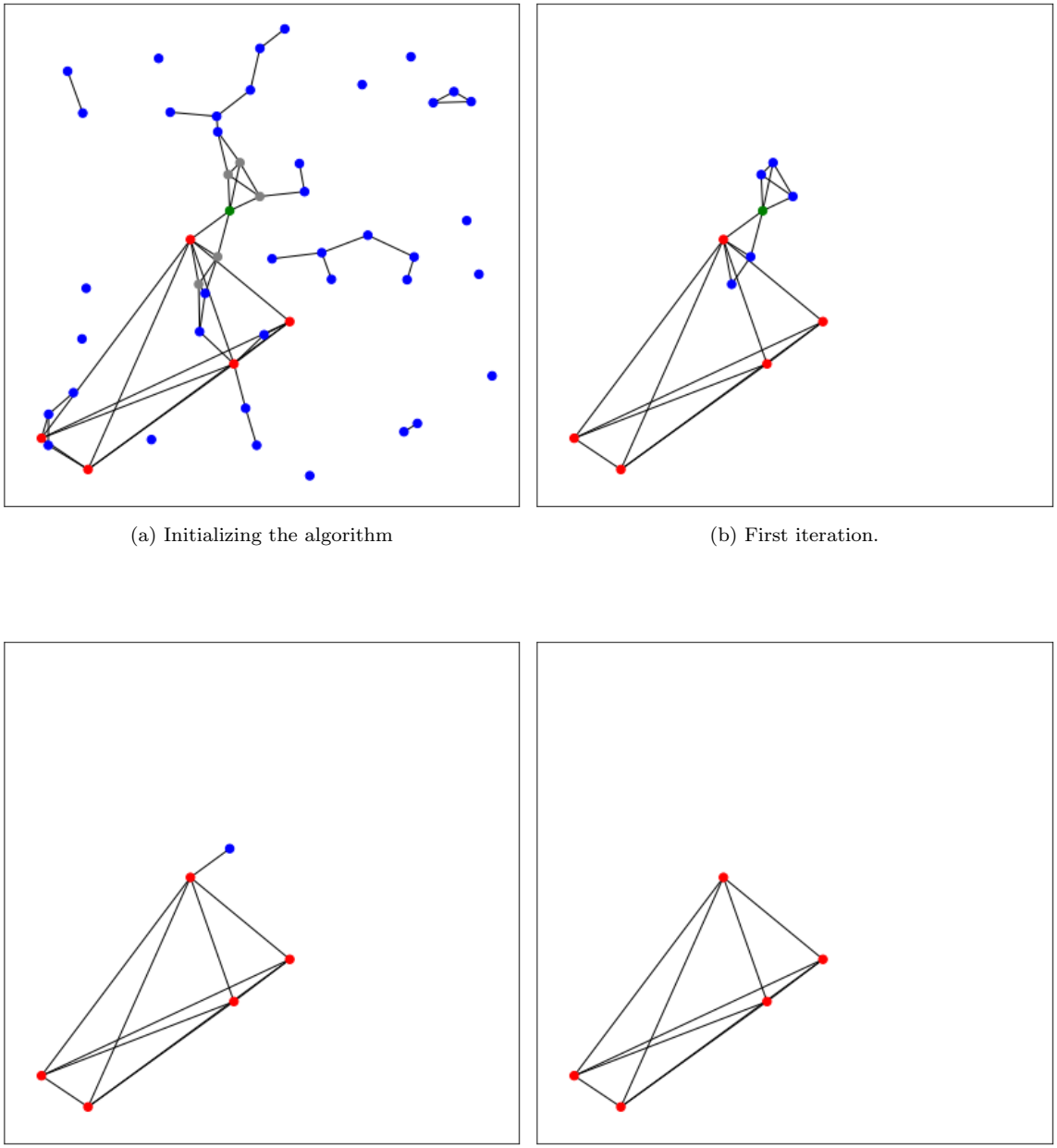
```
Vertex Removal algorithm

input: G(n,r,k)
output: planted clique K
whilst exists i in V, deg(i)<k:
    for all vertices i in V:
        if deg(i)<k-1:
            remove i from graph
    update vertex degree of all vertices left
remaining vertices = K
return K
```

The advantage of this algorithm is that unlike the naive algorithm where the top k vertex degree vertices are taken from graph $G(n, r, k)$, this algorithm does not necessarily fail when the graph $G(n, r, k)$ contains some vertex $v \notin K$ where $deg(v) > deg(u)$ for some $u \in K$. Each iteration of the Vertex Removal algorithm will decrease the vertex degree of at least one vertex, potentially reducing the vertex degree of vertex v below $k - 1$. Another advantage is the fact the no vertex from the planted clique will ever be removed, since any vertex $i \in K$ has a vertex degree of $deg(i) \geq k - 1$ no matter how many low degree vertices are removed. Consider the following example of a small random geometric graph with planted clique. The vertices are marked with four different colours:

1. Red indicates the planted clique vertices
2. Blue indicates vertices with degree $\leq k - 1$, which is removed from the graph.
3. Grey indicates vertices of degree $\geq k - 1$, but have degree smaller than any vertex from the planted clique.
4. Green indicates vertices not from the planted clique that have degree larger equal to the minimal vertex degree of any vertex in the planted clique.

Initializing each iteration of the algorithm, the blue vertices are marked to be removed from the graph. After removal of the blue vertices, the graph is re-evaluated and new colors are assigned to each vertex. This process converts some of the previously green/grey vertices into blue vertices, which can then be removed again next iteration(s) as seen in figure 7.



(a) Initializing the algorithm

(b) First iteration.

(c) Second iteration.

(d) Third iteration.

Figure 7: Full iterations of the Vertex Removal algorithm.

This process is repeated until no more vertices are left of vertex degree $\leq k-1$, which leaves us with the planted clique. In the next section the area of applicability of this algorithm will be discussed in detail.

3.2 Area of applicability

In this section the area of applicability of the vertex removal algorithm will be explained in detail. Before we investigate the applicability in detail we look at some of the properties of the graph with planted clique after applying the algorithm.
Suppose that connected component(s) C_i, $i = 1, 2, ...$ are left after the algorithm terminates. Then the components C_i have the following properties:

1. $|C_i| \geq k$, with equality implying that C_i is a clique of size k.
2. For all vertices $v \in C_i$, $d(v) \geq k-1$.
3. For all vertices $v \in C_i$, $|N(v) \cap C_i| \geq k-1$.

With these properties in mind, we can introduce the area of applicability in Theorem 3, along with a proof on each statement.

Theorem 3. *The vertex removal algorithm recovers the planted clique with high probability if one of the following holds:*

1. $k \geq \epsilon log(n)$ *and* $\frac{1}{\sqrt{n}} \ll r \ll \sqrt{\frac{log(n)}{n}}$.
2. $k \geq \pi n r^2$ *and* $r \gg \sqrt{\frac{log(n)}{n}}$.

Proof. First let us consider what vertices will be left after performing the Vertex Removal algorithm on random geometric graph $G(n, r)$. The structure will consist of one or multiple components $C_i \subseteq V$ that have size $C_i \geq k$, with $|C_i| = k$ implying that component C_i is a clique of size k. We want the probability that the existence of any component not equal to the planted clique $C_i \neq K$ vanishes, so that the algorithm returns the planted clique only. This occurs when the value of k is sufficiently large, with the exact values shown later on.
The Vertex Removal algorithm recovers the planted clique only if the only vertices in the graph left are from the planted clique after the algorithm terminates. The proof will show that this indeed occurs for the given conditions.

Suppose the set of vertices returned by the algorithm is $S \subseteq V$, and let $G[S]$ denote the subgraph induced in $G(n, r)$ by S. When talking about connected components, we talk about connected components in $G[S]$. The following proofs utilise the local density of the number of vertices, so whether we look at the graph after vertex removal $G_r(n, r, k)$ or the induced subgraph $G[S]$, the local vertex density does not change. Two situations are considered that occur when the vertex removal algorithm recovers any vertex outside of the planted clique.

1. Only one component remains that contains some vertices outside of the planted clique after Vertex Removal algorithm terminates.
2. More than one component remains that contain some vertices outside of the planted clique after Vertex Removal algorithm terminates.

The proof will consist of the following sections, by proving each case separately

1. In the proof of case 2 and 3, we always use some vertex outside of the planted clique, thus we first show that we can omit the planted clique vertices.
2. Disprove: Only one component remains that contains some vertices outside of the planted clique after Vertex Removal algorithm terminates.
   (a) Where $k \geq \epsilon \log(n)$ and $\frac{1}{\sqrt{n}} \ll r \ll \sqrt{\frac{log(n)}{n}}$.
   (b) Where $k \geq \pi n r^2$ and $r \gg \sqrt{\frac{log(n)}{n}}$.
3. Disprove: More than one component remains that contain some vertices outside of the planted clique after Vertex Removal algorithm terminates.
   (a) Where $k \geq \epsilon \log(n)$ and $\frac{1}{\sqrt{n}} \ll r \ll \sqrt{\frac{log(n)}{n}}$.
   (b) Where $k \geq \pi n r^2$ and $r \gg \sqrt{\frac{log(n)}{n}}$.

We will now consider each of the three cases above.

Case 1
First of all to show that we can omit the planted clique vertices without affecting the proof:
Let $k = c\log(n)$ where c is a constant larger than 0 and $\frac{1}{\sqrt{n}} \ll r \ll \sqrt{\frac{log(n)}{n}}$. The number of planted clique vertices within the neighbourhood $N(v)$ of a random vertex $v \notin K$ has a distribution of Poisson($c\log(n)r^2$). We have that $r \ll \frac{\log(n)}{n}$, so $c\log(n)r^2 \to 0$; the number of vertices $i \in N(v)$ such that $i \in K$ approaches 0.
For $k \gg \log(n)$ and $\frac{1}{\sqrt{n}} \ll r \ll \sqrt{\frac{\log(n)}{n}}$, we know that no vertex outside of the planted clique $v \notin K$ has such a high vertex degree [9], so the algorithm will always recover the planted clique.

Let $k = cnr^2$ where c is a constant larger than 0 and $r \gg \sqrt{\frac{log(n)}{n}}$. Then the number of planted clique vertices within the neighbourhood $N(v)$ of a random vertex $v \notin K$ has a distribution of Poisson(cnr^4). Since the vertex degree of a vertex $v \notin K$ has a distribution of Poisson($\pi n r^2$) and $r \to 0$, we can say that the portion of vertices in $N(v)$ that belong to the planted clique approaches 0.
For $k \gg nr^2$, we know that no vertex outside of the planted clique $v \notin K$ has such a high vertex degree [9], so the algorithm will always recover the planted clique, and it does so within one iteration.

Case 2a
Only one component remains in the graph after vertex removal terminates. Let this component be C_1, and suppose $k \geq \pi n r^2$ and $\frac{1}{\sqrt{n}} \ll r \ll \sqrt{\frac{log(n)}{n}}$. Consider the original graph $G(n, r)$ before vertex removal. Since $r \ll \sqrt{\frac{log(n)}{n}}$, we know that the graph is disconnected [1]. Let D be an arbitrary connected component in $G(n, r)$, and choose i, j where $i \in D, j \notin D$ such that $d(i, j) \leq d(u, v)$, $\forall u \in D, v \notin D$. Suppose $d(i, j) = x$ $(x > r)$, consider the intersection area $A(x)$ of the circles of radius x around each of the vertices i, j. This intersection has area $(2/3\pi - \sqrt{3}/2)x^2$. We will now show that this area cannot contain any vertex. Suppose that there is a vertex $\exists a \in A(x)$, then vertex a cannot be connected to both i and j, since that implies i and j belong to the same connected

component (assumption i, j disconnected). If a and i are not connected, $d(i, a) < d(i, j)$, which contradicts our selection of i, j. Now suppose that a and i are connected, then $d(a, j) < d(i, j)$ which also contradicts our selection of i, j. Similarly a cannot be connected to, nor disconnected from j. With this, we conclude that $A(x)$ contains no vertices. Since i, j are disconnected $d(i, j) = x > r$, so $\min_{x>r} A(x) \geq (2/3\pi - \sqrt{3}/2)r^2$.

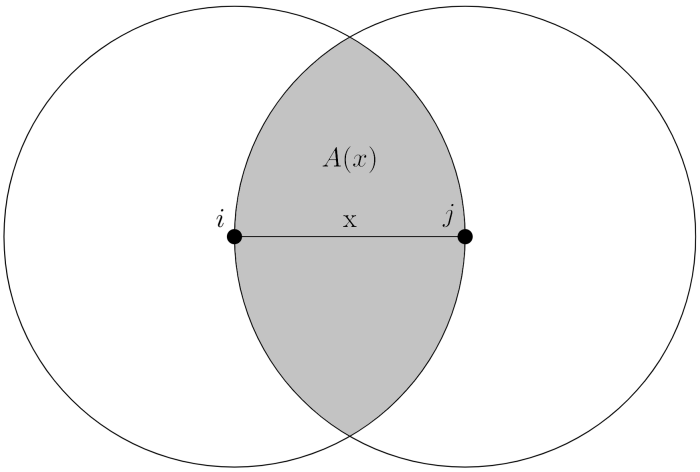


Figure 8: Area A(x)

The same holds for component C_1 in the graph after Vertex Removal algorithm terminates. If C_1 is the only component left after algorithm terminates (excluding the planted clique), it is possible to artificially add a vertex z that is disconnected from C_1 due to the original graph being disconnected. There will always be a vertex $u \in C_1$, such that $d(u, z) \leq d(v, z)$ for all $v \in C_1$ where $d(u, z) \geq r$ (since z disconnected from C_1). Thus vertex u has an empty area in its neighbourhood area of size at least $A(r)$, since no other vertex is closer to the auxiliary vertex z. Now we know that component C_1 must contain a vertex $u \in C_1$ such that all neighbours of u in the component, $N(u) \cap C_1$ fit into a an area of size $\pi r^2 - A(r)$. Since the vertices in $G(n, r)$ follow a Poisson point process, the number of vertices in this area has a distribution of Poisson$((\pi r^2 - A(r))n)$. Let $k \geq \epsilon \log(n)$, and $r \ll \sqrt{\frac{\log(n)}{n}}$. We have that $\pi n r^2 - A(r)n \ll \log(n)$, but for the ease of notation we say that $nr^2 - A(r)n = c\frac{\log(n)}{f(n)}$, where $f(n) = o(\log(n))$ is some monotone increasing function in n and c is some constant larger

than 0, then by Poisson bound [6],

$$P(D_u \geq k) \leq P(D_u \geq \epsilon \log(n)) \tag{43}$$

$$\leq \frac{(e(\pi r^2 - A(r))n)^{\epsilon \log(n)} e^{(-(\pi r^2 - A(r))n)}}{(\epsilon \log(n))^{\epsilon \log(n)}} \tag{44}$$

$$= \frac{(ec\frac{\log(n)}{f(n)})^{\epsilon \log(n)} e^{(-c\frac{\log(n)}{f(n)})}}{(\epsilon \log(n))^{\epsilon \log(n)}} \tag{45}$$

$$= \left(\frac{ec}{\epsilon f(n)}\right)^{\epsilon \log(n)} e^{(-c\frac{\log(n)}{f(n)})} \tag{46}$$

$$\leq \left(\frac{ec}{ce^{3/(\epsilon)}}\right)^{\epsilon \log(n)} e^{(-c\frac{\log(n)}{f(n)})}. \tag{47}$$

Suppose $D_j \sim \text{Poisson}((\pi r^2 - A(r))n)$, then by union bound,

$$\cup_{j=1}^{n} P(D_j \geq k) \leq n \left(\frac{e}{e^{3/(\epsilon)}}\right)^{\epsilon \log(n)} e^{(-c\frac{\log(n)}{f(n)})} \tag{48}$$

$$\leq n \cdot n^{\epsilon} \frac{1}{n^3} e^{(-c\frac{\log(n)}{f(n)})} \tag{49}$$

$$\leq n \cdot n^{\epsilon} \cdot n \frac{1}{n^3} e^{(-c\frac{\log(n)}{f(n)})} \tag{50}$$

$$= n^{2+\epsilon} \frac{1}{n^3} e^{(-c\frac{\log(n)}{f(n)})} \tag{51}$$

$$\xrightarrow[n\to\infty]{} 0. \tag{52}$$

Thus for $k \geq \epsilon \log(n)$, vertex u cannot exist, implying that with high probability, the remaining connected component is the planted clique.

Case 2b

Let $r \gg \sqrt{\frac{\log(n)}{n}}$ and suppose that a singular component C_1 is left after vertex removal that contains some vertices outside of the planted clique. There will be always be empty areas in the graph where no vertices are present after the algorithm terminates, we will prove that the largest circular empty area S_e is larger than a circle of size $\epsilon \pi r^2$. Using the size of empty area S_e, we will show that no singular component C_1 that is not the planted clique can exist after running the Vertex Removal algorithm on the graph $G(n, r, k)$.

Finding large enough circular empty area S_e in the graph

Let us consider the graph $G_r(n, r, k)$, after vertex removal, and let $k \geq \pi n r^2$. Suppose the largest empty circular area is $S_e = \epsilon \pi r^2$, where $\epsilon > 0$ is a small constant. Consider circular areas c_i of size $\epsilon_1 \pi r^2$, with radius $\sqrt{\epsilon_1} r$ where $\epsilon_1 > \epsilon$ such that $c_i > S_e$. We know that within each area c_i, there must be at least one vertex present, since area of c_i is larger than the largest empty circular area S_e in the graph. Consider a larger outer circle $c_{out,i}$ of radius $r + \sqrt{\epsilon_1} r$, such that it covers the neighbourhood of any vertex within circle c_i. The Vertex Removal algorithm terminates when all vertices are of at least degree k, which means that all areas $c_{out,i}$ as shown in figure 9, contain at least k vertices. Within a unit square, the area on which the graph is defined, it is possible to place the following number of circles $c_{out,i}$, such that all $c_{out,i}$ are disjoint as shown in figure 10. Such placement gives us $L_1 = 2(r + \epsilon_1 r)$ and $L_2 = \sqrt{L_1^2 - (1/2L_1)^2} = \sqrt{3 + 3\epsilon_1 + 6\sqrt{\epsilon_1} r}$ with the

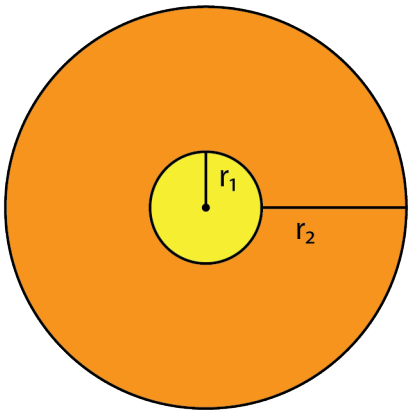


Figure 9: Circle c_i with outer circle $c_{out,i}$, where $r_1 = \sqrt{\epsilon_1}r$ and $r_2 = r$. The entire area will contain at least k vertices in $G_r(n, r, k)$

lengths marked in figure 10, giving us the number of areas $c_{out,i}$ that fit into a unit square:

$$\frac{1}{L_1 L_2} = \frac{1}{2r + 2\sqrt{\epsilon_1}r} \frac{1}{\sqrt{3 + 3\epsilon_1 + 6\sqrt{\epsilon_1}}r} \tag{53}$$

$$= \frac{1}{4r^2 + 4\sqrt{\epsilon_1}r^2 + 4\epsilon_1 r^2}. \tag{54}$$

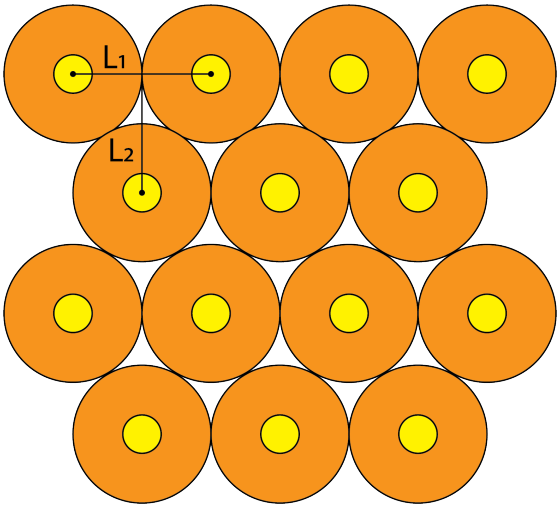


Figure 10: Placement of disjoint areas $c_{out,i}$

Again we know that each one of these areas $c_{out,i}$ contain at least k vertices. The number of

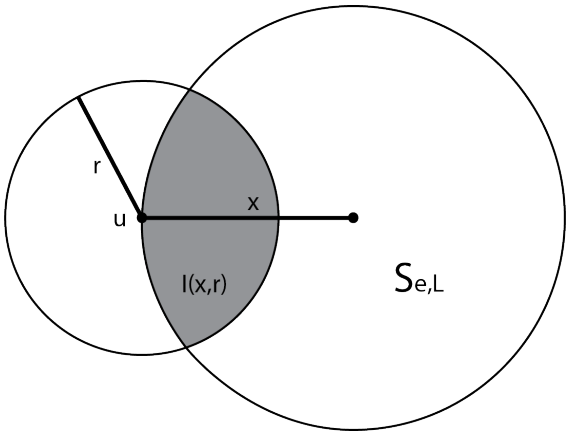


Figure 11: Area $I(x, r)$

vertices in the remaining graph $G_r(n, r, k)$ is then at least

$$\frac{k}{4r^2 + 8\sqrt{\epsilon_1}r^2 + 4\epsilon_1 r^2} = \frac{\pi n r^2}{4r^2 + 8\sqrt{\epsilon_1}r^2 + 4\epsilon_1 r^2} \tag{55}$$

$$\tag{56}$$

By the definition of the Poisson point process, vertices in the graph $G(n, r)$ that have a vertex degree larger than $k = \pi n r^2$ is almost exactly half since k is average degree, we have that

$$\lim_{n\to\infty} P(D_i < \pi n r^2) = 0.5 \text{ as } n \to \infty. \tag{57}$$

Adding the planted clique to the graph $G(n, r)$ will increase the number of vertices with degree larger than $k = \pi n r^2$, since planted clique vertices have vertex degree of $d(i) \geq k | i \in K$. Then, after adding the edges of the planted clique, we have the number of vertices larger than degree k is increased by $1/2k$ since the planted clique vertices are sampled randomly. Thus after running the Vertex Removal algorithm on the graph $G(n, r, k)$, the number of vertices left cannot exceed $1/2n + 1/2k$ by a large number. But since the resulting number of vertices given in equation 55 is much larger than $1/2n + 1/2k$, the assumption that there is no empty area S_e is contradicted, proving that circular empty area S_e exists.

Proving $\exists v \in C_1$ such that $N(v)$ partially overlaps with S_e
Next we want to find a vertex $v \in C_1$ such that there is an empty area of size at least cS_e in its neighbourhood radius, where $c \in (0, 1]$. Let us call the largest empty space S_e that has size at least $\epsilon\pi r^2$.
Suppose no vertex $v \in C_1$ has empty area of size cS_e in its neighbourhood radius. This can only occur when no neighbourhood of any vertex $v \in C_1$ has overlap with S_e. If this is the case, it is possible to expand the radius the circular area S_e until it reaches a vertex $u \in C_1$, call this new larger area $S_{e,L}$. Suppose the vertex u is reached when the empty area has radius of x, we know that $x > r$ since it was assumed that S_e had no overlap with any neighbourhood of vertices $v \in C_1$. Then the neighbourhood area of u contains an empty area $I(x, r)$ as shown in figure 11 Clearly $I(x, r) = c_1\pi r^2 > cS_e = c\epsilon\pi r^2$, where $c, c_1 \in (0, 1]$ and $\epsilon > 0$ is a small constant, thus we found a vertex $u \in C_1$ such that there is an empty area of size at least $cS_{e,L}$ in its neighbourhood radius.

Proving that C_1 cannot exist
Assuming that the neighbourhood of a vertex $v \in C_1$ partially overlaps with S_e, then an empty space of size $c\epsilon\pi r^2$ is present in its neighbourhood, where $c \in (0, 1]$ is a constant. One of the properties of the algorithm is vertex v has degree k after vertex removal. But since we know that an area is empty in its neighbourhood, the degree distribution is given by Poisson$(\pi nr^2(1-c\epsilon))$. Let $k \geq \pi nr^2$ and $r = \sqrt{\frac{\log(n)}{n}} f(n) \gg \sqrt{\frac{\log(n)}{n}}$ where $f(n)$ is any monotone increasing function in n, then the probability that any vertex such a degree distribution has degree larger than k is given by a similar derivation as 43. The only difference here being that the lower bound of k varies with both r and n:

$$P(D_v \geq k) \leq P(D_v \geq \pi nr^2) \tag{58}$$

$$\leq \frac{(e\pi nr^2(1-c\epsilon))^{\pi nr^2} e^{-\pi nr^2(1-c\epsilon)}}{(\pi nr^2)^{\pi nr^2}} \tag{59}$$

Then by the union bound, we can perform the following derivation,

$$\cup_{i=1}^{n} P(D_i \geq k) \leq n \frac{(e\pi nr^2(1-c\epsilon))^{\pi nr^2} e^{-\pi nr^2(1-c\epsilon)}}{(\pi nr^2)^{\pi nr^2}} \tag{60}$$

$$= n(1-c\epsilon)^{\pi nr^2} e^{c\epsilon\pi nr^2} \tag{61}$$

$$= ne^{\log(1-c\epsilon)\pi nr^2} e^{c\epsilon\pi nr^2} \tag{62}$$

$$= ne^{(\log(1-c\epsilon)+c\epsilon)\pi nr^2}. \tag{63}$$

Using the Maclaurin Series $\log(1-x) = -\sum_{i=1}^{\infty} \frac{x^i}{i}$, and substituting $x = c\epsilon$,

$$\cup_{i=1}^{n} P(D_i \geq k) \leq ne^{(\log(1-c\epsilon)+c\epsilon)\pi nr^2} \tag{64}$$

$$= ne^{\left(c\epsilon - \sum_{i=2}^{\infty} \frac{(c\epsilon)^i}{i}\right)\pi nr^2} \tag{65}$$

$$= ne^{\left(-\sum_{i=2}^{\infty} \frac{(c\epsilon)^i}{i}\right)\pi nr^2} \tag{66}$$

$$= ne^{\left(-\sum_{i=2}^{\infty} \frac{(c\epsilon)^i}{i}\right)\pi \log(n) f(n)} \tag{67}$$

$$= n \cdot n^{\left(-\sum_{i=2}^{\infty} \frac{(c\epsilon)^i}{i}\right) f(n)} \tag{68}$$

$$= n^{\left(-\sum_{i=2}^{\infty} \frac{(c\epsilon)^i}{i}\right) f(n)-1} \xrightarrow[n\to\infty]{} 0. \tag{69}$$

With this we have proven that $\cup_{i=1}^{n} P(D_i \geq k) \to 0$ as $n \to \infty$, so that vertex v (vertex with empty area of size cS_e in its neighbourhood area and degree $deg(v) \geq k$) cannot exist in the graph, implying that C_1 cannot exist either.

Case 3a
More than one component remains that contain some vertices outside of the planted clique after Vertex Removal algorithm terminates. Let $k \geq \epsilon \log(n)$ and $\frac{1}{\sqrt{n}} \ll r \ll \sqrt{\frac{log(n)}{n}}$. This proof for this case is similar to the proof of **case 2a**, only here we can omit the part where we choose artificial vertex z that is disconnected from the component. Choose arbitrary component C_1 in $G[S]$, the

subgraph induced in $G(n,r)$ by S, where S is the set of vertices left after vertex removal. Choose vertices i,j where $i \in C_1, j \notin C_1$ such that $d(i,j) \leq d(u,v), \forall u \in C_1, v \notin C_1$. Suppose $d(i,j) = x$, consider the intersection area $A(x)$ of the circles of radius x around each of the vertices i,j. This intersection has area $A(x) = (2/3\pi - \sqrt{3}/2)x^2$. Now we want to show that area $A(x)$ is empty and cannot contain any vertices. Vertex a cannot be connected to both i and j, since that implies i and j are connected. If a and i are not connected, $d(i,a) < d(i,j)$, which contradicts our selection of i,j. If a and i are connected, $d(a,j) < d(i,j)$, which also contradicts our selection of i,j. Similarly a and j cannot be connected nor disconnected. Thus no vertices can exist in $A(x)$. Since i,j are disconnected $d(i,j) = x > r$, it gives us $\min_{x>r} A(x) \geq (2/3\pi - \sqrt{3}/2)r^2$.
We know that vertex i only has neighbours in an area of $\pi r^2 - A(x)$. By Poisson point process, the number of vertices in this area has distribution $D_i \sim \text{Poisson}((\pi r^2 - A(r))n)$. Then following the same derivation as equations 43 to 52, the graph must contain vertex i, but vertex i cannot exist, proving that case 3a does not occur.

Case 3b
More than one component remains that contain some vertices outside of the planted clique after Vertex Removal algorithm terminates. Let $\frac{1}{\sqrt{n}} \ll r \ll \sqrt{\frac{\log(n)}{n}}$ and $k \geq \epsilon \log(n)$, then the proof that vertex i does not exist is exactly the same as the proof of such a vertex not existing in the case of a singular component C_1 left for $\frac{1}{\sqrt{n}} \ll r \ll \sqrt{\frac{\log(n)}{n}}$. Thus in the case of $\frac{1}{\sqrt{n}} \ll r \ll \sqrt{\frac{\log(n)}{n}}$ and $k \geq \epsilon \log(n)$ the vertex removal algorithm recovers the planted clique with high probability.

Let $r \gg \sqrt{\frac{\log(n)}{n}}$ and $k \geq \pi n r^2$, then by Poisson bound

$$P(D_i \geq k) \leq P(D_i \geq \pi n r^2) \tag{70}$$

$$\leq \frac{(e(\pi r^2 - A(r))n)^{\pi n r^2} e^{(-(\pi r^2 - A(r))n)}}{(\pi n r^2)^{\pi n r^2}} \tag{71}$$

$$= \left(1 - \frac{A(r)}{\pi r^2}\right)^{\pi n r^2} e^{A(r)n}. \tag{72}$$

$$\tag{73}$$

By making use of this inequality and the union bound, for $D_j \sim \text{Poisson}((\pi r^2 - A(r))n)$

$$\cup_{j=1}^{n} P(D_j \geq k) \leq n \left(1 - \frac{A(r)}{\pi r^2}\right)^{\pi n r^2} e^{\frac{A(r)}{\pi r^2} \pi n r^2}. \tag{74}$$

$$\tag{75}$$

We know that $A(r) = (2/3\pi - \sqrt{3}/2)r^2 \approx 1.228 r^2$ where $1 - \frac{A(r)}{\pi r^2} \approx 0.609$, so we can use the same derivation as equations 61 to 69, substituting $\frac{A(r)}{\pi r^2}$ for $c\epsilon$, such that

$$\cup_{j=1}^{n} P(D_j \geq k) \leq n \left(1 - \frac{A(r)}{\pi r^2}\right)^{\pi n r^2} e^{\frac{A(r)}{\pi r^2} \pi n r^2} \xrightarrow[n \to \infty]{} 0. \tag{76}$$

$$\tag{77}$$

Thus for $k \geq \pi n r^2$, a vertex where the number of neighbours is distributed by $D_i \sim \text{Poisson}((\pi r^2 - A(r))n)$ cannot have a vertex degree of $deg(i) \geq k$. It is thus proven that C_1 does not exist for the given bounds of r and k, meaning that the Vertex Removal algorithm returns the planted clique only. □

3.3 Complexity

The computation complexity of the Vertex Removal algorithm is different depending on the value of k and r. For k large enough, such that with high probability no vertices in the graph has degree larger equal to k, the algorithm only needs to remove vertices once. In that case the complexity is $O(m)$ or $O_p(n^2r^2)$ to check all vertex degrees once. This occurs if one of the following holds,

1. $k \geq \epsilon log(n)$ and $\frac{1}{\sqrt{n}} \ll r \ll \sqrt{\frac{log(n)}{n}}$ for some $\epsilon > 0$.

2. $k \geq (1+\epsilon)\pi nr^2$ and $r \gg \sqrt{\frac{log(n)}{n}}$ for some $\epsilon > 0$.

The complexity is $O(mn)$ or $O_p(n^3r^2)$, when the algorithm is not guaranteed to terminate within one iteration. This occurs if the following holds,

1. $(1+\epsilon)\pi nr^2 > k \geq \pi nr^2$ and $r \gg \sqrt{\frac{log(n)}{n}}$ for some $\epsilon > 0$.

The complexity is worse when k is around average degree, but for any other values of k, the algorithm typically has quite good worst case complexity. This will be studied more in depth using numerical experiments in the next section.

4 Numerical Experiments

The theory described previously shows the behaviour of the algorithms when n grows to infinity. However this does not tell us much about the performance of the algorithms on finite graphs $G(n, r, k)$ where $n < \infty$. To investigate the practical performance of the neighbourhood algorithm and the Vertex Removal algorithm, numerical experiments will be performed on the two algorithms with varying values of n, r and k.

- $k = \pi n r^2$ is the average degree of a vertex in graph $G(n, r)$.
- $k = \frac{n}{2}$ is a value of order $\theta(n)$ and is important to see if the practical implementation could recover any planted clique at all.
- $k = \frac{n}{\log(n)}$ is a value that is still large, but significantly smaller than n.
- $k = \sqrt{n}$ is a significantly smaller value than $\frac{n}{\log(n)}$, but a clique of this size should still be possible to be recovered by all algorithms for $r = 2\sqrt{\frac{\log(n)}{n}}$.
- $k = \log(n)$ and $k = 2\log(n)$ are very small values for k that are often not possible to recover accurately theoretically. However algorithms could work better in practice than in theory, so these values of k are worth investigating as well.
- $r = \frac{1}{\sqrt{n}}$ is the smallest value of r that we will use in this paper.
- $r = 2\sqrt{\frac{\log(n)}{n}}$ is a value of r around the connectivity threshold [1].
- $r = \frac{1}{n^{1/4}}$ is a value that is significantly larger than the connectivity threshold.

The numerical experiments will focus on two aspects, the accuracy of the algorithms, and the run time of the algorithms. The accuracy for the two algorithms were each calculated differently.
The accuracy of the neighbourhood algorithm is given by the number of planted clique vertices recovered divided by k. This algorithm always returns k vertices, and at failure the k vertices returned contains some non clique vertices.
The accuracy of the Vertex Removal algorithm is given by the size of the planted clique divided by the total amount of vertices recovered by the algorithm, the accuracy is calculated like this because the vertices in the graph that the algorithm returns ($G_r(n, r, k)$) may exceed size k, but the graph always contains the entire planted clique as a subgraph.
To ensure that each algorithm is tested thoroughly, the algorithms will be tested for 20 different graphs for every value of k and r. The size of the graphs that the algorithms will be tested on is $n = 1000$. Taking larger values for n will give us more accurate results, but the time to generate the graphs increases as n^2.

4.1 Accuracy

In table 1 and 2 we present the results for the cases listed above. The cells of each table represent the average accuracy over 20 instances.

k \ r	$\frac{1}{\sqrt{n}}$	$\sqrt{\frac{\log(n)}{n}}$	$\frac{1}{n^{1/4}}$
πnr^2	0.0608	0.0190	0.499
$\frac{n}{2}$	1.0	1.0	1.0
$\frac{n}{\log(n)}$	1.0	1.0	1.0
$\sqrt{n}$	1.0	0.609	0.1209
$2\log(n)$	1.0	0.00769	1.0

Table 1: Accuracy neighbourhood algorithm run for 20 iterations with $n = 1000$

k \ r	$\frac{1}{\sqrt{n}}$	$\sqrt{\frac{\log(n)}{n}}$	$\frac{1}{n^{1/4}}$
πnr^2	0.0039	0.943	1.0
$\frac{n}{2}$	1.0	1.0	1.0
$\frac{n}{\log(n)}$	1.0	1.0	1.0
$\sqrt{n}$	1.0	1.0	0.031
$2\log(n)$	1.0	0.013	0.013

Table 2: Accuracy Vertex Removal algorithm run for 20 iterations with $n = 1000$

The accuracy obtained from the numerical experiments on the neighbourhood algorithm in section 2 also correspond to the theoretical results. The algorithm showed high accuracy when tested on its theoretical area of applicability. As expected, unlike the Vertex Removal algorithm, the neighbourhood algorithm did not perform well when the planted clique was of average degree size (πnr^2). The accuracy obtained from the numerical experiments on the Vertex Removal algorithm in section 3 corresponds exactly to the theoretical results. The algorithm Vertex Removal algorithm had high accuracy in the numerical experiments when tested on its theoretical area of applicability. Due to the nature of the Vertex Removal algorithm, one is always able to verify the correctness of the results in practice very quickly; anytime the algorithm terminates with a graph consisting of more than k vertices, it is known that the algorithm has failed to recover only the planted clique vertices. However even at failure, it is certain that the resulting graph always contains the planted clique as a subgraph.

4.2 Run-time

The running time of algorithms do not always correlate to the worst case theoretical complexity. In this section the theoretical complexity of the algorithms will be further analyzed by observing the algorithms' run-time in practice. As seen in sections 2.3 and 3.3, the complexity of the Neighbourhood and Vertex Removal algorithm is:

1. neighbourhood algorithm: Worst case complexity $O(m)$ or $O_p(n^2r^2)$
2. Vertex Removal algorithm: Worst case complexity of $O(mn)$ or $O_p(n^3r^2)$

The run-time of the two algorithms are measured with graphs of size $n = 100, 500, 1000$ and the given values for r and k.

4.2.1 neighbourhood algorithm

k \ r	$\frac{1}{\sqrt{n}}$	$2\sqrt{\frac{\log(n)}{n}}$	$\frac{1}{n^{1/4}}$
πnr^2	0.01396	0.02293	0.01697
$n/2$	0.004981	0.01298	0.06690
$\frac{n}{\log(n)}$	0.001994	0.003983	0.005004
$\sqrt{(n)}$	0.006009	0.01305	0.001992
$2\log(n)$	0.005019	0.093945	0.002995

Table 3: Neighbourhood algorithm run-time in seconds,$n = 100$, experiments with high accuracy marked in green.

k \ r	$\frac{1}{\sqrt{n}}$	$\sqrt{\frac{\log(n)}{n}}$	$\frac{1}{n^{1/4}}$
πnr^2	0.08976	0.15662	0.13970
$n/2$	0.04500	0.32419	0.08274
$\frac{n}{\log(n)}$	0.02083	0.03785	0.08774
$\sqrt{(n)}$	0.02292	0.03291	0.02190
$2\log(n)$	0.02088	0.07274	0.02095

Table 4: Neighbourhood algorithm run-time in seconds,$n = 500$, experiments with high accuracy marked in green.

k \ r	$\frac{1}{\sqrt{n}}$	$\sqrt{\frac{\log(n)}{n}}$	$\frac{1}{n^{1/4}}$
πnr^2	0.0608	0.1286	1.2696
$\frac{n}{2}$	0.1096	0.07668	0.4157
$\frac{n}{\log(n)}$	0.3591	0.5484	2.4274
$\sqrt{(n)}$	1.3124	1.5048	1.7881
$2\log(n)$	3.4448	0.03696	21.0058

Table 5: Neighbourhood algorithm run-time in seconds,$n = 1000$, experiments with high accuracy marked in green.

The computational complexity of the neighbourhood algorithm is

1. if $k > \omega(n, r)$: $\mathcal{O}\left(n^2k^2 + n^3\right)$ or $\mathcal{O}\left(mk^2 + mn\right)$.
2. if $k < \omega(n, r)$: $\mathcal{O}\left(n^2\omega(n, r)^2 + n^3\right)$ or $\mathcal{O}\left(mk^2 + mn\right)$.

Generally speaking it is observed that the run-time increases as r increases, but sometimes an increased value for r causes the algorithm to run much faster. When the graph has a large number of vertex pairs with exactly $k - 2$ the algorithm tends to fail at recovering the correct subset of planted clique vertices. This in turn causes the algorithm to terminate fast, since a random vertex pair with $k - 2$ vertices can quickly be found.

4.2.2 Vertex Removal algorithm

k \ r	$\frac{1}{\sqrt{n}}$	$2\sqrt{\frac{\log(n)}{n}}$	$\frac{1}{n^{1/4}}$
πnr^2	0.01396	0.01066	0.01191
$n/2$	0.01096	0.01394	0.02700
$\frac{n}{\log(n)}$	0.00199	0.003983	0.00500
$\sqrt(n)$	0.00601	0.01305	0.00199
$2\log(n)$	0.00502	0.09394	0.00300

Table 6: Vertex Removal Algorithm runtime $n = 100$, experiments with high accuracy marked in green.

k \ r	$\frac{1}{\sqrt{n}}$	$2\sqrt{\frac{\log(n)}{n}}$	$\frac{1}{n^{1/4}}$
πnr^2	0.05973	0.14164	0.11067
$n/2$	0.04296	0.06582	0.12454
$\frac{n}{\log(n)}$	0.02083	0.03785	0.08774
$\sqrt(n)$	0.02292	0.03291	0.02190
$2\log(n)$	0.02088	0.07274	0.02095

Table 7: Vertex Removal Algorithm run-time in seconds,$n = 500$, experiments with high accuracy marked in green.

k \ r	$\frac{1}{\sqrt{n}}$	$2\sqrt{\frac{\log(n)}{n}}$	$\frac{1}{n^{1/4}}$
πnr^2	0.10773	0.12431	0.34506
$n/2$	0.11977	0.13767	0.28920
$\frac{n}{\log(n)}$	0.05599	0.11182	0.30021
$\sqrt(n)$	0.05288	0.09471	0.06304
$2\log(n)$	0.05203	0.1920	0.05437

Table 8: Vertex Removal Algorithm run-time in seconds,$n = 1000$, experiments with high accuracy marked in green.

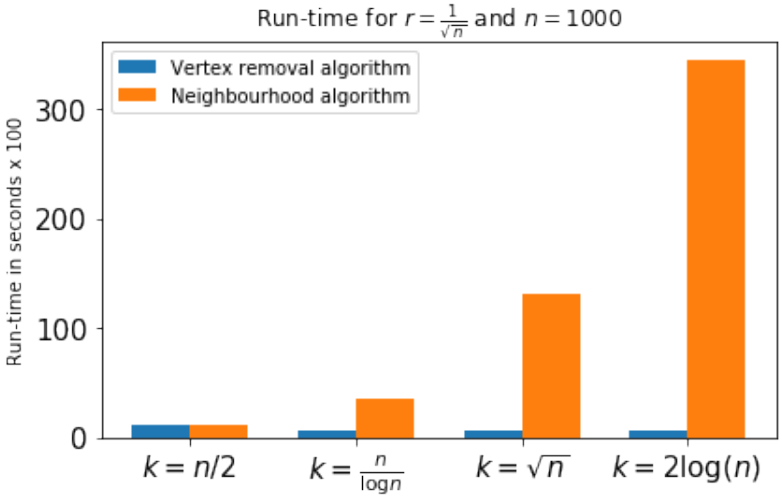
The theoretical worst case complexity of the Vertex Removal algorithm is of order $O(n^3r^2)$, so according to the theory it is expected that the run-time of the numerical experiments increases as n increases. That is indeed what is observed in the results of the numerical experiments; for set functions of k and r, increasing the graph size n will always lead to a longer run-time. As the graph size n increases however we do observe that the run-time does not nearly increase with an order of n^3, this shows that in practice the algorithm runs faster than expected.

The only times when the run-time does not increase, with an increased r, is when the algorithm

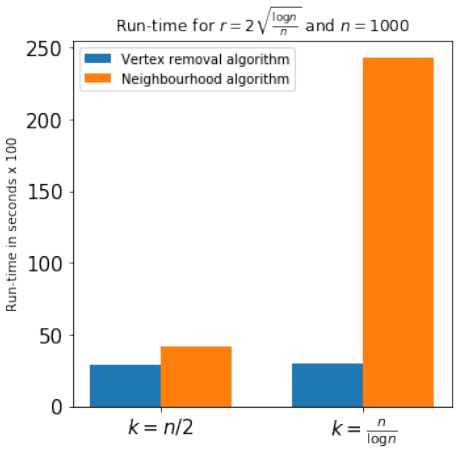
fails to find the planted clique. Due to the nature of the Vertex Removal algorithm, the algorithm terminates when no more vertices can be removed from the graph. When the algorithm fails to recover the planted clique it means that a large number ($> k$) of vertices are left within the graph. This causes the algorithm to terminate early, since not many vertices had to be removed. This is an excellent property of the algorithm, since one is able to quickly verify whether the algorithm fails.

4.2.3 Comparison

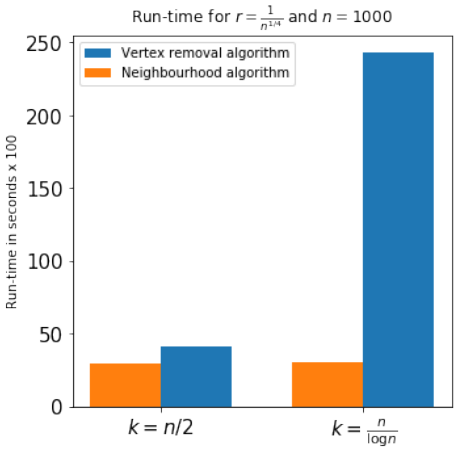
As seen in sections 3.3 and 2.3, the theoretical worst case complexity of both algorithms are quite similar. But as mentioned before, the complexity of the Vertex Removal algorithm rarely reaches the worst case. We observe this phenomenon when comparing the run-time of both algorithms; restricting ourselves to only comparing the run-time of experiments when both algorithms have high accuracy, the neighbourhood algorithm almost always has a much higher run-time than the Vertex Removal algorithm as seen in figure 11. This is no surprise since the majority of the time (for $k \geq (1+\epsilon)\pi nr^2$), the complexity of the Vertex Removal algorithm is lower.



(a)



(b)



(c)

Figure 11: Comparison run-time Neighbourhood algorithm and Vertex removal algorithm, for cases where both algorithms have high accuracy.

5 Conclusion and further research

5.1 Conclusion

The goal of this paper was to find methods in order to recover a planted clique from random geometric graphs of size n, where n is a large number tending towards infinity. This was not a well research problem in the literature, since past research mostly dealt with the setting of recovering a planted clique in Erdős–Rényi Graphs. In this paper two algorithms are introduced for solving the problem. The Neighbourhood algorithm 2 utilizes the number of common neighbours between vertex pairs to find a pair of planted clique vertices. What is remarkable about this algorithm is that it is able to recover the planted clique for a wide range planted clique size k, most notably the case of fixed k. The downside of this algorithm lies in its high computational complexity. The Neighbourhood algorithm was implemented in python to test its practical performance. The Neighbourhood algorithm was tested thoroughly within its theoretical area of applicability to evaluate its practical performance, the results were promising with high accuracy tested in 20 iterations each time. The Neighbourhood algorithm was also tested in an area outside of its theoretical applicability for $k = \pi nr^2$ (average vertex degree), but more often than not the algorithm failed to recover the planted clique.

The Vertex Removal algorithm 3 is a method that utilizes the property of a large clique having high degree vertices in order to recover the planted clique. Unlike the Neighbourhood algorithm the Vertex Removal is most likely cannot recover a planted clique of fixed size. However the Vertex Removal algorithm is able to recover the planted clique of size average vertex degree (πnr^2) under certain conditions of r, which the Neighbourhood algorithm was unable to do so. The Vertex Removal algorithm was tested for its practical applicability, with the results indicating high accuracy in its theoretical area of applicability. The Vertex Removal algorithm has a worst case complexity of $O_p(mn)$, but in practice that rarely applies as the worst case complexity is only achieved in graphs with exceptionally specific setups. As observed from the practical experiments, the run-time of the Vertex Removal algorithm is significantly lower than the run-time of the Neighbourhood algorithm independent of the chosen values of r, k and n.

5.2 Further research

We have a number of suggestions for future research. First of all notice that there is no extensive statements for negative results. Finding all values of r and k in which the algorithms do not work may aid in determining the precise theoretical area of applicability of the algorithms. Subsequently, there may exist improvements for the area of applicability described by Theorem 10. So far we have proved that the Vertex Removal algorithm recovers the planted clique with high probability for certain conditions of k and r, but that does not mean that these are the absolute best bounds.

It might also be interesting to find the average complexity of the algorithms, as the practical results indicated that the run-time does not increase as fast as the worst case complexity does. Also to get the best results for practical experimentation, it is recommended to test on even larger graphs than $n = 1000$.

The Vertex Removal algorithm could have an application in planted clique recovery in Erdős–Rényi graphs. The Vertex Removal algorithm is an algorithm based on the fact that little vertices exist in the graph with exceptionally high vertex degree as $n \to \infty$. The vertex degrees of vertices in an Erdős–Rényi graph have a binomial distribution, which converges to the Poisson distribution if np is fixed or $p \to 0$. The vertex degree distribution is then similar to the vertex degree distribution of

the random geometric graph.

A potential method for planted clique recovery in random geometric graphs that was briefly experimented with was graph embedding. Graph embedding uses the adjacency matrix to visually reconstruct the graph on some kind of plane. It could then be possible to find the planted clique using the visualization. We didn't manage to find promising results with short experimentation.

References

[1] Mathew D. Penrose. *Random Geometric Graphs*. Oxford University Press, 2003.

[2] Colin Mcdiarmid, Tobias Müller. *On the chromatic number of random geometric graphs*. Combinatorica, November 2011, Volume 31, Issue 4, pp 423–488.

[3] Michel Goemans. *Chernoff bounds, and some applications*. Lecture Notes MIT, 2015.

[4] N. Alon, M. Krivelevich, B. Sudakov. *Finding a Large Hidden Clique in a Random Graph*. Proceedings of the Eighth International Conference "Random Structures and Algorithms" held August 4–9, 1997, in Poznan, Poland, October - December 1998.

[5] L. Kučera. *Expected complexity of graph partitioning problems*. Discrete Applied Mathematics, 1995, Volume 57 (2–3): 193–212.

[6] M. Mitzenmacher, E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge, UK: Cambridge University Press, ISBN 978-0-521-83540-4

[7] William Henry Burt. *Territoriality and Home Range Concepts as Applied to Mammals*. Journal of Mammalogy, Volume 24, Issue 3, 17 August 1943, Pages 346–352

[8] Seth A. Myers, Aneesh Sharma, Pankaj Gupta, and Jimmy Lin. *Information Network or Social Network? The Structure of the Twitter Follow Graph* . Journal of Mammalogy, Volume 24, Issue 3, 17 August 1943, Pages 346–352

[9] K. Avrachenkov, A. Bobu. *Planted clique recovery in random geometric graphs (manuscript in preparation)*. January 17, 2020