The Complexity of Finding Dense Subgraphs in Graphs with Large Cliques

by

Cameron Seth

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2021

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The GapDensest-k-Subgraph(α) problem (GapDkS(α)) is defined as follows: given a graph G and parameters α, k, distinguish between the case that G contains a k-clique, and the case that every k-subgraph of G has density at most α.

GapDkS(α) is a natural relaxation of the standard k-Clique problem, which is known to be NP-complete. For α very close to 1, the GapDkS(α) problem is equivalent to the k-Clique problem, and when α is very close to 0 the GapDkS(α) problem can easily be solved in polynomial time. However, despite much work on both the algorithmic and hardness front, the exact k and α parameter values for which GapDkS(α) can be solved in polynomial time are still unknown. In particular, the best polynomial-time algorithms can solve GapDkS(α) when α is an inverse polynomial in the number of vertices n, but there have been no NP-hardness results beyond the trivial result.

This thesis attempts to understand the GapDkS(α) problem better by studying the case when k is restricted to be linear in n (where n is the number of vertices in G). In particular, we survey the GapDkS(α) algorithms and hardness results that can be best applied to this restriction in an attempt to determine the threshold for when the problem becomes NP-hard. With some modifications to the algorithms and proofs, we produce algorithms and hardness results for the GapDkS(α) problem with $k = \Omega(n)$. Informally, we show that GapDkS(α) with $k = \Omega(n)$:

- is NP-hard to solve when $\alpha > 1 - O(n^{-\delta})$ for any constant $\delta > 0$,
- can be solved in randomized polynomial time when $\alpha < 1 - \Omega(1/\log n)$, and
- can be solved in polynomial time when $\alpha < 1 - \epsilon$ for any constant $\epsilon > 0$.

In addition, we study the connection between GapDkS(α) and MaxClique, and show that despite having strong hardness results for MaxClique, reductions from MaxClique do not give strong hardness bounds for GapDkS(α).

Acknowledgements

Firstly I would like to thank my supervisor Eric Blais. I have learned a lot over the last two years and I wouldn't have been able to do it without his guidance. I would also like to thank my lab mates Nathan, Renato and Abhinav for the many helpful discussions. Finally I would like to thank my family (Ravi, Marisa, Natasha, Micaala, Mom and Dad) and my girlfriend Tori for all their support over the years.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 GapDensest-k-Subgraph(α) Problem

The NP-complete problems are a class of decision problems that appear across many areas of computer science. The boolean satisfiability problem was the first problem shown to be NP-complete, proven independently by Cook [10] and Levin [26]. Shortly after, many other problems were shown to be NP-complete, including a list of 21 problems by Karp [21]. These problems are all related in that if we could solve any one of them in polynomial time, we could solve all of them in polynomial time. However, despite much effort, there have been no polynomial-time algorithms found for these problems, and it is conjectured that these problems require superpolynomial time to solve. This is the famous P vs NP conjecture.

The k-Clique problem is a standard NP-complete problem [21]. The input to k-Clique is a graph G and a parameter k, and the output is YES if G contains a clique of size k, and NO otherwise.

In an effort to understand the NP-complete class better, we can study relaxations of NP-complete problems. One way to relax the k-Clique problem is to only demand that the output is NO when all k-subgraphs of G have density lower than some input parameter. We call this decision problem the GapDensest-k-Subgraph(α) problem, shortened to GapDkS(α).

GapDkS(α): The input to GapDkS(α) is a graph G, on n vertices, and parameters $k \leq n, \alpha \in (0,1)$. The goal is to distinguish between the case that G contains a clique of size k, and the case that every k-subgraph of G has density less than α (where the density is defined as the number of edges in the induced subgraph divided by $\binom{k}{2}$).

The above definition is often referred to as the *decision version of* Densest-k-Subgraph (DkS) *with perfect completeness*. The "perfect completeness" refers to the fact that the YES instances are graphs containing k-subgraphs of density 1. There is a more general problem

without perfect completeness where the YES instances have density greater than some other input parameter. We give more details on these related problems in Section 2.3.

A natural question is: for what values of k and α can the GapDkS(α) problem be solved in polynomial time, and for what values is it NP-complete?

There are two trivial bounds for GapDkS(α). If $\alpha < \frac{k/2}{\binom{k}{2}}$ then the problem can be solved in polynomial time by simply checking if the graph contains $k/2$ edges. On the other extreme, if $\alpha > 1 - \frac{1}{\binom{k}{2}}$ then the problem is NP-complete because it is equivalent to distinguishing if the graph contains a k-clique or not.

For general k the answer to the above question is still widely unknown. The best polynomial-time algorithms can solve GapDkS(α) when $\alpha < n^{-\epsilon}$ for any constant $\epsilon > 0$ [16, 28], but there have been no NP-hardness results beyond the trivial hardness result.[1] To gain insight into this question, this thesis studies the $k = \Omega(n)$ restriction of the GapDkS(α) problem. In particular, this thesis aims to answer the following question.

Question. *With the restriction that $k = \Omega(n)$, for what values of α can the* GapDkS(α) *problem be solved in polynomial time, and for what values is it* NP*-Complete?*

There are a few reasons why this restriction of GapDkS(α) makes sense to study. First, under the $k = \Omega(n)$ restriction, the GapDkS(α) problem is still NP-hard for general α. However, it is believed that the $k = \Omega(n)$ instances are slightly easier to solve than the general case. In particular, there are polynomial-time algorithms for GapDkS(α) with $k = \Omega(n)$ and constant α [16], but, assuming the Exponential Time Hypothesis (ETH), there are no polynomial-time algorithms to solve GapDkS(α) for general k and constant α [9, 27].

1.2 Main Results

This thesis covers both the algorithmic and hardness bounds on the GapDkS(α) problem when $k = \Omega(n)$.

On the algorithmic side, we review a deterministic algorithm by Feige and Seltser that illustrates the use of the greedy strategy [16].

Theorem 1.1 (Feige, Seltser 1997)**.** *For any constant $\epsilon > 0$,* GapDkS(α) *with $k = \Omega(n)$ and $\alpha < 1 - \epsilon$ can be solved in* ***deterministic*** *polynomial time. More generally, for any k there exists an algorithm which runs in time $n^{O((\log(n/k)+1)/\epsilon)}$.*

The algorithm from Theorem 1.1 is the best known deterministic polynomial-time algorithm for GapDkS(α) when $k = \Omega(n)$, however, if we allow randomization we can solve GapDkS(α) for a larger α parameter range. We show this by modifying a Clique testing algorithm by Goldreich *et al.* [18] to get the following theorem.

[1] See Section 3.1 and 4.1 for more details on the history of the problem.

Theorem 1.2. *For any constant $\epsilon > 0$, $\textsc{GapD}k\textsc{S}(\alpha)$ with $k = \Omega(n)$ and $\alpha < 1 - \epsilon/\log n$ can be solved in **randomized** polynomial time. More generally, for any k there exists an algorithm which runs in time $n^{O((n/k)\log(n/k)/\epsilon)}$.*

On the hardness side, we adapt Manurangsi's hardness proof [27] for the general $\textsc{GapD}k\textsc{S}(\alpha)$ problem to get a NP-hardness result for $\textsc{GapD}k\textsc{S}(\alpha)$ with $k = \Omega(n)$.

Theorem 1.3. *For any constant $\delta \in (0, 1)$, there exists a parameter $k = \Omega(n)$ and a constant $c > 0$ such that $\textsc{GapD}k\textsc{S}(\alpha)$ is NP-hard when $\alpha > 1 - c/n^{\delta}$.*

We note that there remains a gap between the algorithmic and hardness bounds on α from Theorem 1.2 and 1.3.

1.3 Structure of this Thesis

Chapter 2 reviews some notation and definitions we use throughout the thesis.

In Chapter 3 we give a brief history of the algorithmic techniques for DkS and study the most applicable algorithms to $\textsc{GapD}k\textsc{S}(\alpha)$ with $k = \Omega(n)$. We utilize the greedy strategy to get a simple algorithm for $\textsc{GapD}k\textsc{S}(\alpha)$, and show how it can be extended to prove Theorem 1.1. We also show the relationship between property testing of the k-Clique problem, and the $\textsc{GapD}k\textsc{S}(\alpha)$ problem when $k = \Omega(n)$. We modify and strengthen the algorithm from Goldreich *et al.* [18] to prove Theorem 1.2.

In Chapter 4, we give a brief history of the hardness results for DkS. We motivate the proof techniques for Theorem 1.3 by proving two simple hardness results for $\textsc{GapD}k\textsc{S}(\alpha)$. We prove Theorem 1.3 by modifying a hardness proof of Manurangsi [27]. We complete the chapter by demonstrating the limitations of the reduction technique used.

In Chapter 5 we study the relationship between $\textsc{GapD}k\textsc{S}(\alpha)$ and another common relaxation of k-Clique called MaxClique. We show that we can use the hardness results of MaxClique to get a hardness result for $\textsc{GapD}k\textsc{S}(\alpha)$, but this method cannot prove stronger hardness results than Theorem 1.3.

Finally, in Chapter 6 we mention some open problems and questions for future work.

Chapter 2

Preliminaries

In this section we review some definitions and notations that are used throughout the thesis.

2.1 Graphs

Throughout this thesis we only consider finite, undirected, unweighted graphs without self loops or multi-edges. We denote graphs by $G = (V, E)$ and use n to denote the number of vertices in the graph. We use $\deg(v)$ to denote the degree of a vertex. When the graph is not clear, we use $\deg_G(v)$ to specify the graph for which the degree is calculated.

A graph $G = (V, E)$ has a clique of size k if there exists a subset $K \subseteq V$, of size k, such that for every two distinct $v_1, v_2 \in K$, $(v_1, v_2) \in E$. We use the terminology *k-subgraph* of G to mean an induced subgraph of size k within G.

For any graph $G = (V, E)$, the *density* of G is defined as $|E|/\binom{n}{2}$. The density ranges from 0 to 1, where 0 corresponds to an empty graph, and 1 corresponds to a complete graph.

In Chapter 4 we study graph homomorphisms on bipartite graphs. A *graph homomorphism* from G_1 to G_2 is a function mapping vertices of G_1 to vertices of G_2 such that adjacent vertices in G_1 become adjacent vertices in G_2. In other words, a function $f : V_1 \to V_2$ is a homomorphism from $G_1 = (V_1, E_1)$ to $G_2 = (V_2, E_2)$ if for every pair of vertices $u, v \in V_1$, we have that $(u, v) \in E_1 \Rightarrow (f(u), f(v)) \in E_2$. Since we are dealing with graphs without self loops, this means that adjacent vertices in G_1 must map to distinct vertices in G_2. However, we note that if two vertices are not adjacent in G_1, then it is possible they map to the same vertex in G_2.

For any $t \in \mathbb{N}$, let $K_{t,t}$ denote the complete bipartite graph with t vertices per part. We specifically study homomorphisms from $K_{t,t}$, and in particular, a labelled copy of $K_{t,t}$.

2.2 CSPs and PCPs

This thesis uses the Constraint Satisfaction Problem (CSP) to prove hardness results for GapDkS(α). The input to CSP is a formula ϕ. A *formula* is a set of M constraints $\{C_1, \ldots, C_M\}$ over a set of N variables $\{x_1, \ldots, x_N\}$, where each variable can take a value from some alphabet Σ. Each constraint C_i is a function that takes as input some subset of the variables, and produces a boolean value as output. We say ϕ is *satisfiable* if there is an assignment to the variables such that every constraint evaluates to true. The output of CSP is YES if ϕ is satisfiable, and NO otherwise.

In this thesis we study two restrictions of the CSP problem.

- An instance to 2CSP$_\Sigma$ is a formula where each constraint depends on at most two variables, and the variables come from the alphabet Σ.
- An instance to 3Sat is a formula where each constraint depends on at most three variables, and each constraint is a disjunction ("OR" function) of literals, where a literal is a variable or its negation. Each variable is on the boolean alphabet. For example, $(x_1 \vee x_2 \vee \neg x_4)$ is a 3Sat constraint. In the 3Sat problem, constraints are referred to as *clauses*, and formulas are often written as $C_1 \wedge C_2 \wedge \ldots \wedge C_M$.

For any CSP instance ϕ, we define the *value* of ϕ, denoted val(ϕ), to be the maximum fraction of constraints that can be simultaneously satisfied over all possible assignments to the variables. In particular, if ϕ is satisfiable then val(ϕ) $= 1$, and if ϕ is unsatisfiable then val(ϕ) < 1.

Both 2CSP$_\Sigma$ and 3Sat are known to be NP-hard. In other words, it is NP-hard to distinguish between the cases that val(ϕ) $= 1$ and val(ϕ) < 1. In a series of works, culminating in the Probabilistically Checkable Proof (PCP) Theorem [5, 4], it was shown that it is NP-hard to distinguish between the cases that val(ϕ) $= 1$ and val(ϕ) $< 1 - \delta$ for some constant $\delta > 0$. Chapter 4 uses results from Dinur's simplified proof of the PCP theorem [11].

The Exponential Time Hypothesis (ETH) is a conjecture stating that 3Sat cannot be decided in $2^{o(N)}$ time [20]. ETH is commonly used as an assumption in DkS hardness proofs.

2.3 Related Problems

There are a number of related problems that we use to prove results about GapDkS(α).

GapDkS(β, α): GapDkS(β, α) is the same problem as GapDkS(α) but without perfect completeness. In other words, the input to GapDkS(β, α) is a graph G and a parameter k, and the output is YES if G contains a k-subgraph of density at least β, and NO if every k-subgraph of G has density less than α. When $\beta = 1$ then we recover the GapDkS(α) problem.

Many algorithms are meant for the more general GapDkS(β, α), but they can still be applied to GapDkS(α).

Densest-k-Subgraph Optimization (DkS): DkS is defined as an optimization problem as opposed to a decision problem. In the optimization problem, the input is a graph G and a parameter k, where G is guaranteed to contain a k-clique, and the goal is to find a k-subgraph with high density. Again, this is the perfect completeness version of the problem because G is guaranteed to contain a k-clique.

In the optimization view, we study algorithms that have some density guarantee on the output. All algorithms in this thesis aim to approximate the optimization version of DkS, and these algorithms can be used to solve the decision version GapDkS(α). This connection is summarized in the lemma below.[1]

Lemma 2.1. *Let A be a polynomial-time algorithm for* DkS *optimization that, on input G, k, is guaranteed to find a k-subgraph with density at least δ. Then there exists a polynomial-time algorithm for* GapDkS(δ).

Proof. Let G be an instance of GapDkS(δ). Consider a new algorithm A' that runs $A(G, k)$ and returns YES if $A(G, k)$ returns a k-subgraph with density larger than δ, and returns NO otherwise.

If G contains a k-clique then $A(G, k)$ is guaranteed to find a subgraph with density at least δ, and hence A' returns YES. If every k-subgraph of G has density strictly less than δ, then it's not possible for $A(G, k)$ to find a k-subgraph with density δ. In this case the algorithm returns NO. In either case A' correctly decides GapDkS(δ), and its running time is the same as the running time of algorithm A. □

Similarly, hardness results for the optimization version follow immediately from proving hardness for the decision version. In other words, if GapDkS(δ) is NP-hard, then it is NP-hard to solve DkS with density guarantee δ.

In the literature, the density guarantee is called an approximation ratio. An algorithm is said to have *approximation ratio* $\eta > 1$ if it finds an estimate to the solution that is at least as large as the optimal value divided by η. In the case of DkS, an algorithm that has an approximation ratio η is guaranteed to find a k-subgraph with density larger than $1/\eta$.

MaxClique: MaxClique is a second common relaxation of the clique problem. Given input G, instead of finding a subgraph with maximal density, the goal is to find a clique of maximal size (the size of the maximal clique is called the *clique number*). This problem has been studied extensively. On the hardness side it has been shown that it is NP-hard to approximate to $n^{1-\epsilon}$ for any constant $\epsilon > 0$ [19, 32]. This has been strengthened to $n^{1-\epsilon}$ for some subconstant ϵ by Khot [22, 24]. The best known algorithms give approximations of roughly $O\left(\frac{n}{\log^3 n}\right)$ in polynomial time [13].

[1] We note that Lemma 2.1 applies to both deterministic and randomized algorithms.

Property Testing Clique: In this problem the input is a graph G and parameters $\rho, \epsilon > 0$ and the goal is to distinguish, with high probability, between the case that G has a clique of size ρn and the case that G is ϵ-far from having a clique of size ρn (where ϵ-far means that at least ϵn^2 edges must be added to contain a clique) [18]. While the focus of property testing algorithms is generally to have sublinear running time and query complexity, the problem is very similar to GapDkS(α). If a k-subgraph is α-dense, then it is $\frac{(1-\alpha)\binom{k}{2}}{n^2}$-far from having a clique of size k.

2.4 Useful Bounds

Throughout this thesis we use the following inequalities to simplify expressions.

Exponential Approximations: $1 + x \leq e^x$ for any $x \in \mathbb{R}$ and $1 - \frac{y}{2} \geq 2^{-y}$ for any $0 \leq y \leq 1$.

Binomial Approximation: $\left(\frac{n}{\ell}\right)^\ell \leq \binom{n}{\ell} \leq \left(\frac{en}{\ell}\right)^\ell$ for any $1 \leq \ell \leq n$.

Arithmetic Mean and Geometric Mean (AM-GM) Inequality: $xy \leq \left(\frac{x+y}{2}\right)^2$.

Markov's Inequality: If X is a non-negative random variable and $a > 0$ then

$$\Pr[X \geq a] \leq \frac{\mathbb{E}[X]}{a}.$$

We use $\exp(x)$ and e^x interchangeably, and use $\log(x)$ to represent the base two logarithm.

Chapter 3

DkS Algorithms

In this chapter we first give a history of DkS algorithms. Then we study two algorithmic techniques that can be applied to the GapDkS(α) problem. In Section 3.2 we study the greedy strategy. In particular, we give a basic greedy algorithm and demonstrate how it can be improved to the following deterministic algorithm (which was proven by Feige and Seltser [16]).

Theorem 1.1 (Feige, Seltser 1997)**.** *For any constant $\epsilon > 0$,* GapDkS(α) *with $k = \Omega(n)$ and $\alpha < 1 - \epsilon$ can be solved in* ***deterministic*** *polynomial time. More generally, for any k there exists an algorithm which runs in time $n^{O((\log(n/k)+1)/\epsilon)}$.*

In section 3.3 we apply property testing algorithms for k-Clique to GapDkS(α). We demonstrate how the original Clique testing algorithm of Goldreich *et al.* [18] can be used directly to solve GapDkS(α). Further, we simplify the algorithm and improve the proof to get the randomized algorithm in Theorem 1.2. The algorithm in Theorem 1.2 finds a higher density k-subgraph than Theorem 1.1, however, Theorem 1.2 is a randomized algorithm, whereas Theorem 1.1 is deterministic.

Theorem 1.2. *For any constant $\epsilon > 0$,* GapDkS(α) *with $k = \Omega(n)$ and $\alpha < 1 - \epsilon/\log n$ can be solved in* ***randomized*** *polynomial time. More generally, for any k there exists an algorithm which runs in time $n^{O((n/k)\log(n/k)/\epsilon)}$.*

As mentioned in Section 2.3, to prove Theorem 1.1 and Theorem 1.2 we find algorithms to approximate the DkS optimization problem. Then the theorems follow directly by applying Lemma 2.1.

3.1 History of DkS Algorithms

Most algorithms for the DkS problem aim to solve the more general DkS problem without perfect completeness. In this problem, the input is a graph G and a parameter k, without

the promise that G contains a k-clique, and the output is a k-subgraph with some density guarantee relative to the optimal (see Section 2.3 for more details).

The best polynomial-time algorithms for the general DkS problem without perfect completeness achieve polynomial approximation ratios. In particular, algorithms with approximation ratios of roughly $n^{0.3885}, n^{1/3}, n^{1/4}$ for DkS without perfect completeness are given by Kortsarz and Peleg [25], Feige *et al.* [15], and Bhaskara *et al.* [7] respectively.

All of the above algorithms can be used to solve GapDkS(α) (with perfect completeness) for any k, however the α parameter must be less than an inverse polynomial. For example, the $n^{1/4}$ approximation algorithm of Bhaskara *et al.* [7] corresponds to $\alpha < n^{-1/4}$.

There are some algorithms for DkS without perfect completeness that can do better for GapDkS(α) with $k = \Omega(n)$ because they have a dependence on k in the approximation ratio. Algorithms by Feige and Langberg [14], Sritsav and Wolf [29], and Asahiro *et al.* [6] use various techniques to achieve approximation ratios of roughly n/k in polynomial time. These algorithms correspond to polynomial-time algorithms for GapDkS(α) when $\alpha < k/n$, which is a constant when $k = \Omega(n)$.

The algorithms that are best suited to the $k = \Omega(n)$ case come from results by Arora *et al.* [3] and Feige and Seltser [16]. The algorithm by Arora *et al.* is specifically designed for the $k = \Omega(n)$ case (but not necessarily perfect completeness), whereas the algorithm by Feige and Seltser has an explicit dependence on k in the running time of the algorithm. Both of these algorithms approximate DkS to any constant approximation ratio, and when $k = \Omega(n)$ they can be used to solve GapDkS(α) with perfect completeness when $\alpha < c$ for any constant $c \in (0, 1)$ in polynomial time.

The algorithm by Feige and Seltser [16] is specifically designed to solve the perfect completeness case of DkS. In addition to giving any constant approximation ratio in polynomial time when $k = \Omega(n)$, it provides the best polynomial-time approximation (with ratio n^{ϵ} for any constant ϵ) for DkS with perfect completeness on general k. The algorithm uses a greedy strategy which has similarities with some of the other algorithms above [15, 6]. For these reasons we study the greedy strategy and the full proof of the Feige and Seltser algorithm in Section 3.2.

The algorithm of Arora *et al.* [3] (as well as some of the algorithms above [29, 14]) utilize the technique of relaxing the DkS problem to a quadratic integer program. Similar to the technique of Goemans and Williamson for MaxCut [17], they observe that by letting $\{x_1, \ldots, x_n\}$ be boolean values corresponding to whether each vertex is in the k-subgraph, the DkS problem is equivalent to solving the following quadratic integer program:

$$\begin{aligned} \text{maximize} \quad & \sum_{\{i,j\} \in E} x_i x_j \\ \text{subject to} \quad & \sum_i x_i = k, \\ & x_i \in \{0, 1\}. \end{aligned}$$

Written this way, there are many techniques that can be used to approximate the problem,

including semi-definite programming (SDP) [29, 14] or smooth integer programming [3].

Finally, in a completely separate area, the property testing paper of Goldreich, Goldwasser and Ron studies the testing problem for Clique [18]. In this problem the input is a graph G and parameters ρ and ϵ, where ρ and ϵ are constants. The goal is to distinguish between the following cases with probability $\geq 2/3$:

- G has a ρn-clique, and
- G is ϵ-far from having a ρn-clique,

where ϵ-far means that G is missing at least ϵn^2 edges from having a clique of size ρn.

The algorithm by Goldreich *et al.* can be used directly to study the GapDkS(α) when $k = \Omega(n)$. We study this algorithm in Section 3.3.

We also note that the algorithm by Goldreich *et al.* has some similarities with work by Suzuki and Tokuyama [30], however the latter were focused on the bipartite DkS problem and their algorithm only applies to GapDkS(α) when $\alpha < 1/2$.

3.2 Greedy Algorithms for DkS

The greedy algorithms utilize the fact that vertices with high degree are likely part of a dense subgraph. To illustrate this point, we make the observation that removing the lowest degree vertex never decreases the density of a graph. The following lemma comes from the work of Feige and Seltser [16], and is useful in the analysis of the algorithms in Sections 3.2.1 and 3.3.1.

Lemma 3.1. *Let G be a graph with density δ. Let v_{min} be the lowest degree vertex of G, and let G' be the same graph as G but with v_{min} removed. Then the density of G' is at least δ.*

Proof. Let $G = (V, E)$ be a graph on n vertices and $|E|$ edges. Let the density of G be δ (in other words $|E| = \delta\binom{n}{2}$). Let $v_{min} \in V$ be the lowest degree vertex in G, and let $G' = (V', E')$ be the graph by taking G and removing v_{min}. Observe that $\deg(v_{min}) \leq \frac{2|E|}{n}$. Hence, $|E'| \geq |E| - \frac{2|E|}{n} = \delta\binom{n}{2}\left(1 - \frac{2}{n}\right)$. So, the density of G' is

$$\frac{|E'|}{\binom{n-1}{2}} \geq \frac{\delta\binom{n}{2}\left(1 - \frac{2}{n}\right)}{\binom{n-1}{2}} = \delta. \qquad \square$$

Building on Lemma 3.1, we can construct a simple greedy algorithm to solve GapDkS(α) with $\alpha = \frac{k}{2n}$.

Proposition 3.2. *Let G be a graph on n vertices that contains a clique of size k, where $2 \leq k \leq n$. There exists a polynomial-time algorithm to find a k-subgraph of G with density at least $\frac{k}{2n}$.*

Proof. Start with G, and at each step remove the lowest degree vertex until we are left with k vertices. This runs in $O(n^3)$ time because we can calculate the degree of each vertex in time $O(n^2)$, and the algorithm iterates at most n times.

To bound the density of the resulting k-subgraph, observe that the k-clique in G corresponds to k vertices with degree $\geq k-1$. Hence, as we remove the lowest degree vertices one by one, at some point we reach a graph where all vertices have degree $\geq k-1$. Call this intermediate graph G', and let n' be the number of vertices in G'. So, the density of G' is at least

$$\frac{(k-1)n'/2}{\binom{n'}{2}} = \frac{k-1}{n'-1} \geq \frac{k}{2n},$$

where the last inequality is because $n'-1 \leq n$ and $k-1 \geq k/2$ so long as $k \geq 2$.

Finally, we can repeatedly apply Lemma 3.1 until we end up with a graph of size k with density $\geq \frac{k}{2n}$. □

Proposition 3.2, combined with Lemma 2.1, demonstrates that when k is linear in n, there exists a polynomial time algorithm to solve GapDkS(α) when α is smaller than the constant $\frac{k}{2n}$. In Section 3.2.1 we outline an extension of the simple greedy algorithm that can solve GapDkS(α) for any constant $\alpha < 1$.

3.2.1 Feige and Seltser Algorithm

Feige and Seltser gave a deterministic algorithm to approximate the DkS optimization problem [16]. This algorithm utilizes the greedy approach in one of the main steps of the algorithm. We can use this algorithm to prove Theorem 1.1.

Theorem 1.1 (Feige, Seltser 1997)**.** *For any constant $\epsilon > 0$, GapDkS(α) with $k = \Omega(n)$ and $\alpha < 1 - \epsilon$ can be solved in **deterministic** polynomial time. More generally, for any k there exists an algorithm which runs in time $n^{O((\log(n/k)+1)/\epsilon)}$.*

Proof. Let $\epsilon > 0$ be any constant. Consider the algorithm by Feige and Seltser [16] in Figure 3.1.

Claim 3.3 (Correctness)**.** *If G contains a k-clique then FeigeSeltser(G, k, ϵ) returns a k-subgraph with density at least $1 - \epsilon$.*

Before proving the claim fully, we give an intuitive argument. If all the vertices in G have high degree, then G is a dense graph. So, we can apply Lemma 3.1 to show that repeatedly removing the lowest degree vertex (line 20 in the algorithm) does not decrease the density. If there are vertices with low degree, then we recurse on the two cases G_v and G_{-v}. Since G is guaranteed to have a k-clique then at least one of the two cases must contain a k-clique.

Algorithm 1: FeigeSeltser(G, k, ϵ)

```
1  if k > n then
2  |  return the empty graph ;
3  else if k = n and G is not a k-clique then
4  |  return the empty graph ;
5  else if k = n and G is a k-clique then
6  |  return G ;
7  else
8  |  let v ∈ V be the lowest degree vertex in G ;
9  |  let h = deg(v) ;
10 |  if h < (1 − ε)n then
11 |  |  let G_v to be the induced subgraph by v and all of its neighbours ;
12 |  |  let G' =FeigeSeltser(G_v, k, ε) ;
13 |  |  if G' is non-empty then
14 |  |  |  return G' ;
15 |  |  else
16 |  |  |  let G_{−v} be G with v removed ;
17 |  |  |  return FeigeSeltser(G_{−v}, k, ε) ;
18 |  |  end
19 |  else
20 |  |  Repeatedly remove the lowest degree vertex from G until we have a graph G'
   |  |    of size k ;
21 |  |  return G' ;
22 |  end
23 end
```

Figure 3.1: The Feige and Seltser DkS Algorithm [16].

Proof of Claim 3.3. We first observe that on any input G (regardless of if G contains a k-clique) the algorithm can only return an empty graph or a k-subgraph with density at least $1-\epsilon$. This can be shown by examining the two places where the algorithm stops and returns a non-empty graph. In line 6 the algorithm returns a k-clique, and in line 20 the algorithm returns a k-subgraph with density $1-\epsilon$ (by Lemma 3.1). All other "return" statements in the algorithm return an empty graph or are a recursive call.

We now prove the claim using induction on the number of vertices. Let G be any graph on n vertices.

Base Cases: The base cases are when $n \leq k$ (lines 1-6 in the algorithm). In any of these cases, if G contains a k-clique then the algorithm returns it. This corresponds to line 3 in the algorithm.

Inductive Hypothesis: Assume Claim 3.3 holds for all graphs of size $n' < n$.

Inductive Conclusion: If $h < (1-\epsilon)n$ then at least one of G_v, G_{-v} is guaranteed to have a k-clique. If G_v contains a k-clique then the FeigeSeltser(G_v, k, ϵ) recursive call finds a k-subgraph with density at least $1-\epsilon$ by the Inductive Hypothesis. If G_v does not contain a k-clique, then FeigeSeltser(G_v, k, ϵ) can only return an empty graph or a k-subgraph with density $\geq (1-\epsilon)$. If it returns empty then we recurse on G_{-v}. In this case we know that G_{-v} must have the k-clique and so the FeigeSeltser(G_{-v}, k, ϵ) recursive call finds a k-subgraph with density at least $1-\epsilon$ by the Inductive Hypothesis. Finally, if $h \geq (1-\epsilon)n$ then G has density at least $(1-\epsilon)$, and so by Lemma 3.1 the algorithm finds a k-subgraph of density $1-\epsilon$. □

Claim 3.4 (Running Time)**.** *FeigeSeltser(G, k, ϵ) runs in time $n^{O\left(\frac{\log(n/k)+1}{\epsilon}\right)}$, where n is the number of vertices in G.*

Proof of Claim 3.4. In the worst case the algorithm makes recursive calls until reaching the base cases of $n \leq k$. To find the running time we analyze the recurrence. Let $T(n)$ be the number of steps to run FeigeSeltser(G, k, ϵ) on a graph of size n. Recursing on G_v results in at most $T((1-\epsilon)n+1)$ steps because v has degree $< (1-\epsilon)n$. For simplicity in the analysis we upper bound $T((1-\epsilon)n+1) \leq T((1-\epsilon/2)n)$, which is true when ϵ is a constant. Recursing on G_{-v} results in at most $T(n-1)$ steps.

All other steps take at most n^c for some constant c. The constant does not matter for the theorem, but for a worst case analysis we can show all other steps can be done in $O(n^3)$. In particular, checking the base cases and calculating the lowest degree vertex requires $O(n^2)$. Line 20 of the algorithm requires $(n-k) \cdot O(n^2) = O(n^3)$ time to reach a base case, totally $O(n^2) + O(n^3) = O(n^3)$ time.

Putting it all together we get the following recurrence:

$$T(n) \leq O(n^c) + T(n-1) + T((1-\epsilon/2)n).$$

To simplify the recurrence we can repeatedly expand the $T(n-i)$ term until we reach the base case of $k \geq n$:

$$\begin{aligned} T(n) &\leq O(n^c) + T(n-1) + T((1-\epsilon/2)n) \\ &\leq O(n^c) + T(k) + (n-k)\left[O(n^c) + T((1-\epsilon/2)n)\right] \\ &= O(n^{c+1}) + (n-k)T((1-\epsilon/2)n). \end{aligned}$$

We can expand the new recurrence $T(n) \leq O(n^{c+1}) + (n-k)T((1-\epsilon/2)n)$ until we reach the base case of $k \geq n$. This gives us the bound of $T(n) \leq O(n^{c+1+x})$ where x is the number of times we need to repeat until we reach the base case. This happens when $(1-\epsilon/2)^x n \leq k$. Simplifying this we get

$$x \geq \frac{\log(n/k)}{\log(1/(1-\epsilon/2))} = O\left(\frac{\log(n/k)}{\epsilon}\right).$$

Selecting $x = O\left(\frac{\log(n/k)+1}{\epsilon}\right)$ suffices, giving a running time of $n^{O\left(\frac{\log(n/k)+1}{\epsilon}\right)}$. □

Hence the FeigeSeltser algorithm runs in polynomial time when $k = \Omega(n)$ and $\epsilon = \Omega(1)$, and approximates the DkS optimization problem with density guarantee $1-\epsilon$. Applying Lemma 2.1 completes the proof of Theorem 1.1. □

3.3 Property Testing Algorithm for DkS

The property testing paper of Goldreich *et al.* provides a testing algorithm for the k-Clique problem [18]. In the process, they prove the following intermediate step.

Theorem 3.5 (Theorem 7.1 of [18])**.** *Let G be a graph on n vertices and let ρ be a constant. If G contains a clique of size ρn then for any $\epsilon > 0$ there exists a $\exp\left(O\left(\frac{\log(1/\epsilon)\rho}{\epsilon^2}\right)\right) \cdot n$ time algorithm to find, with probability at least $2/3$, a ρn-subgraph that is missing at most ϵn^2 edges from being a clique.*

Theorem 3.5 provides an algorithm for GapDkS(α).

Corollary 3.6. *There exists a randomized polynomial time algorithm for* GapDkS(α) *when $k = \Omega(n)$ and $\alpha < 1 - \Omega\left(\sqrt{\frac{\log\log n}{\log n}}\right)$.*

Proof. On input G and $k = \Omega(n)$, where G is guaranteed to have a k-clique, let $\rho = k/n$ and $\epsilon = \Omega\left(\sqrt{\frac{\log\log n}{\log n}}\right)$. Apply Theorem 3.5 to find a subgraph of size $\rho n = k$ that is missing at most ϵn^2 edges from being a clique. Then the density of this subgraph is at least

$$1 - \frac{\epsilon n^2}{\binom{\rho n}{2}} = 1 - \frac{2\epsilon n}{\rho(\rho n - 1)} \geq 1 - \Omega\left(\sqrt{\frac{\log\log n}{\log n}}\right).$$

Also observe that the algorithm runs in time

$$\exp\left(O\left(\frac{\log(1/\epsilon)\rho}{\epsilon^2}\right)\right)\cdot n = \exp\left(O\left(\frac{\rho\log(\sqrt{\log n}/\sqrt{\log\log n})\log n}{\log\log n}\right)\right)\cdot n = n^{O(1)}.$$

Applying Lemma 2.1 converts the DkS optimization algorithm to an algorithm for GapDkS(α). □

Corollary 3.6 can solve GapDkS(α) for a larger range of α than the FeigeSeltser algorithm. However, with some modifications to the algorithm and proof, we can get an even better algorithm that solves GapDkS(α) when $k = \Omega(n)$ and $\alpha < 1 - \Omega\left(\frac{1}{\log n}\right)$ in randomized polynomial time. We study the modified algorithm and proof in Section 3.3.1.

3.3.1 Modified Clique Testing Algorithm

In this section we strengthen Theorem 3.5 to prove the following theorem.

Theorem 3.7. *Let G be a graph on n vertices. If G contains a k-clique then for any $\epsilon > 0$ there exists a $\exp\left(O\left(\frac{(n/k)\log(n/k)}{\epsilon}\right)\right)\cdot n^2$ time algorithm to find, with probability at least $2/3$, a k-subgraph with density at least $1-\epsilon$.*

We prove Theorem 3.7 below, but first observe that if we select $\epsilon = \frac{\epsilon'}{\log n}$, for any constant $\epsilon' > 0$, and $k = \Omega(n)$, then by applying Lemma 2.1 we prove Theorem 1.2.

Theorem 1.2. *For any constant $\epsilon > 0$, GapDkS(α) with $k = \Omega(n)$ and $\alpha < 1 - \epsilon/\log n$ can be solved in **randomized** polynomial time. More generally, for any k there exists an algorithm which runs in time $n^{O((n/k)\log(n/k)/\epsilon)}$.*

Proof of Theorem 3.7. Let G be a graph on n vertices with a k-clique. Consider the algorithm in Figure 3.2, which is a simplified version of the algorithm from Goldreich *et al.* [18].

Claim 3.8 (Running Time)**.** *Let G be a graph on n vertices. Then CliqueTester(G,k,ϵ) runs in time $\exp\left(O\left(\frac{(n/k)\log(n/k)}{\epsilon}\right)\right)\cdot n^2$.*

Proof of Claim 3.8. Each iteration of the loop takes $O(n^2)$ time to select and calculate all the degrees of the vertices in $T(U_i)$. So, the run time is:

$$O\left((n/k)^t n^2\right) = O\left((n/k)^{\frac{12n}{\epsilon k}} n^2\right) = \exp\left(O\left(\frac{(n/k)\log(n/k)}{\epsilon}\right)\right)\cdot n^2.$$ □

Claim 3.9 (Correctness)**.** *Let G be a graph on n vertices with a k-clique. With probability $2/3$, CliqueTester(G,k,ϵ) returns a k-subgraph with density at least $1-\epsilon$.*

Algorithm 2: CliqueTester(G, k, ϵ)

Let $t = \frac{12n}{\epsilon k}$;
for $i = 1$ *to* $2(n/k)^t$ **do**
  Select t vertices uniformly at random with replacement to create a set U_i ;
  Let $T(U_i)$ be the set of vertices in G that are adjacent to every $v \in U_i$;
  Let $C(U_i)$ be the set of k vertices with highest degree in $T(U_i)$ (if there are less than k vertices in $T(U_i)$ then ignore this U_i and continue) ;
end
return $C(U_i)$ with the highest density ;

Figure 3.2: A Modified Clique Testing Algorithm for DkS

Before proving the claim we give an intuitive argument. First, we repeat the loop enough times so that one set U_i is completely contained within the clique C. For this specific U_i, the clique is contained in $T(U_i)$ because every vertex in U_i is connected to every vertex in the clique. If there are no other vertices in $T(U_i)$ then $C(U_i)$ certainly has high density, however, there may be many other vertices in $T(U_i)$.

Let $\overline{C} = V \setminus C$. For any $v \in \overline{C}$ let k_v be the number of vertices that v is adjacent to in C. In other words, let $k_v = |\{c \in C : (v, c) \in E\}|$. The intuitive idea is that if k_v is small then, for a random $U \subset C$, v is unlikely to be connected to all of U and so it is unlikely that $v \in T(U)$. If k_v is large then v could be selected in $T(U)$, but because it is connected to lots of vertices in C then v could still be part of a dense subgraph. In particular, we want to argue that the vertices of C have high degree within $T(U)$, and so when we select the highest degree vertices to $C(U)$ we find a dense subgraph. We formalize this intuition below.

Proof of Claim 3.9. Let $G = (V, E)$ contain a clique C of size k. First observe that the probability that a random selection of t vertices, with replacement, is contained within the clique C is $(k/n)^t$. In other words, for any $1 \leq i \leq 2(n/k)^t$,

$$\Pr[U_i \not\subset C] = 1 - (k/n)^t \leq e^{-(k/n)^t}.$$

By repeating the loop $2(n/k)^t$ times, we find that

$$\Pr\left[\bigwedge_{i=1}^{2(n/k)^t} U_i \not\subset C\right] = \prod_{i=1}^{2(n/k)^t} \Pr[U_i \not\subset C] \leq e^{-2} \leq \frac{1}{6}.$$

Hence, with probability at least 5/6 the algorithm will find a $U_i \subset C$. For the rest of the proof we focus on this specific set $U \subset C$, and for this U we can assume the vertices are drawn from C randomly.

Since $U \subset C$ then $C \subseteq T(U)$ because every vertex in the clique connects to every vertex in U. As mentioned above, let $\overline{C} = V \setminus C$ and, for any $v \in \overline{C}$, let $k_v = |\{c \in C : (v, c) \in E\}|$.

For each $v \in \overline{C}$, the probability that v is connected to a random vertex in C is $\frac{k_v}{k}$. Hence, v is selected in $T(U)$ with probability $\left(\frac{k_v}{k}\right)^t$ because U is t vertices at random (with replacement) from C. If v is selected in $T(U)$, then it contributes $k - k_v$ *missing edges* to the sum of the degrees of the vertices of C.

So, in expectation, for any $c \in C$, we can calculate the expected degree of c within $T(U)$ as

$$\mathbb{E}[\deg_{T(U)}(c)] = |T(U)| - 1 - \frac{\sum_{v \in \overline{C}} \left(\frac{k_v}{k}\right)^t (k - k_v)}{k} = |T(U)| - 1 - \sum_{v \in \overline{C}} \left(\frac{k_v}{k}\right)^t \left(1 - \frac{k_v}{k}\right).$$

Let $x = \frac{k_v}{k}$ so that the term inside the summation is $x^t(1-x)$. We can find the maximum of this expression by differentiating and finding the zeros:

$$0 = \frac{d}{dx} x^t(1-x) \Rightarrow x = \frac{t}{t+1}.$$

Given this maximal value for $x = \frac{k_v}{k}$, the term inside the summation can be upper bounded by $x^t(1-x) \leq \left(\frac{t}{t+1}\right)^t \frac{1}{t+1} \leq \frac{1}{t}$. Substituting this expression into the expectation calculation, we find that for any $c \in C$

$$\mathbb{E}[\deg_{T(U)}(c)] \geq |T(U)| - 1 - \frac{|\overline{C}|}{t}.$$

Next, we need to calculate the expected degree of any vertex selected in $C(U)$. We make the following two observations.

1. If a vertex $c \in C$ is selected to be in $C(U)$, then $\deg_{C(U)}(c)$ is smaller than $\deg_{T(U)}(c)$ by at most $|T(U)| - |C(U)|$. In other words, $\mathbb{E}[\deg_{C(U)}(c)] \geq |C(U)| - 1 - \frac{|\overline{C}|}{t}$. This is because when the algorithm creates $C(U)$ it effectively removes $|T(U)| - |C(U)|$ vertices from $T(U)$.
2. Since $C(U)$ is selected to be the k vertices with highest degree in $T(U)$, then for any $v \in C(U)$ the above bound $\mathbb{E}[\deg_{C(U)}(v)] \geq |C(U)| - 1 - \frac{|\overline{C}|}{t}$ also holds (even if v is not from the clique C).

Putting it all together, the expected number of edges in $C(U)$ is at least $k\left(k - 1 - \frac{|\overline{C}|}{t}\right)/2$. Hence, in expectation, the density of $C(U)$ is at least

$$\frac{k\left(k - 1 - \frac{|\overline{C}|}{t}\right)/2}{\binom{k}{2}} = 1 - \frac{|\overline{C}|}{t(k-1)} \geq 1 - \frac{2n}{tk},$$

since $|\overline{C}| \leq n$ and $k - 1 \geq k/2$.

Finally, we need to show that with high probability the density of $C(U)$ is larger than $1-\epsilon$. We use Markov's inequality to convert the expected value into a probability.

Let the density of $C(U)$ be a random variable δ, and let $\overline{\delta} = 1 - \delta$. Observe that $\overline{\delta}$ is a non-negative random variable with expectation $\mathbb{E}[\overline{\delta}] = \frac{2n}{tk}$. Applying Markov's inequality, we find that $\Pr[\overline{\delta} \geq a] \leq \frac{\mathbb{E}[\overline{\delta}]}{a}$ for any $a > 0$. Selecting $a = 6\,\mathbb{E}[\overline{\delta}]$ and substituting $t = \frac{12n}{\epsilon k}$, we find that $\Pr[\overline{\delta} \geq \epsilon] \leq \frac{1}{6}$. In other words, with probability at least $\frac{5}{6}$ the density of $C(U)$ is at least $1-\epsilon$.

Finally, the probability that the algorithm finds a set $U_i \subset C$ and the density of $C(U_i) \geq 1-\epsilon$ is at least $\left(\frac{5}{6}\right)^2 \geq \frac{2}{3}$. □

Claims 3.8 and 3.9 combined prove that CliqueTester(G, k, ϵ) has the desired runtime and density guarantee. This completes the proof of Theorem 3.7. □

Chapter 4

GapDkS(α) Hardness when $k = \Omega(n)$

In this chapter we first give a summary of the previous DkS hardness results. Then we apply Manurangsi's proof techniques [27] to the GapDkS(α) problem when $k = \Omega(n)$. With a few small modifications to the original proof, we prove the following theorem.

Theorem 1.3. *For any constant $\delta \in (0, 1)$, there exists a parameter $k = \Omega(n)$ and a constant $c > 0$ such that* GapDkS(α) *is* NP*-hard when $\alpha > 1 - c/n^{\delta}$.*

4.1 Brief History of DkS Hardness

Most hardness results for the DkS problem focus on the more general case of DkS without perfect completeness. In this case the problem is conjectured to be NP-hard to approximate to a polynomial ratio [27], but it has not been proven that a constant approximation is NP-hard. However, there have been many results that rule out constant approximations under various other assumptions [12, 23, 2].

There are two results that apply to the perfect completeness case of GapDkS(α). Braverman *et al.* proved that assuming ETH there are no polynomial-time algorithms to solve GapDkS(α) when α is larger than some constant [9]. Manurangsi strengthened the result by Braverman *et al.* to show that assuming ETH, there are no polynomial-time algorithms to solve GapDkS(α) when α is nearly inverse polynomial. The results by Manurangsi and Braverman *et al.* use a similar reduction from the CSP problem. We study the reduction in the proof of Theorem 1.3.

4.2 Warmup

Before proving Theorem 1.3, we go through a simple reduction from 2CSP$_\Sigma$ to GapDkS(α) to demonstrate the techniques. Proposition 4.1 is a proof of the trivial hardness result for

GapDkS(α). The proposition proves that it is NP-hard to distinguish between graphs with a k-clique and graphs without a k-clique. Proposition 4.3 demonstrates how to strengthen the reduction technique to get a non-trivial hardness result for GapDkS(α).

Proposition 4.1. *The* GapDkS(α) *problem with* $k = \Omega(n)$ *is* NP*-hard when* $\alpha > 1 - \frac{1}{\binom{k}{2}}$*, where* n *is the number of vertices in the input graph.*

Proof. It is known that 2CSP$_\Sigma$ is NP-hard when $|\Sigma| \geq 3$ because it is equivalent to the graph coloring problem [21]. Let ϕ be a instance of 2CSP$_\Sigma$ on N variables and M constraints where $|\Sigma|$ is a constant. Then, in general, it is NP-hard to distinguish between:

- $\text{val}(\phi) = 1$ (ϕ is satisfiable), and
- $\text{val}(\phi) \leq 1 - \frac{1}{M}$ (ϕ is unsatisfiable).

Consider the following procedure to convert ϕ into a graph $G_\phi = (V_\phi, E_\phi)$.

1. For every variable in ϕ, add a vertex for every possible assignment from Σ to that variable. So the number of vertices in G_ϕ is $n = |\Sigma| \cdot N$. We label each vertex with the corresponding single variable assignment.
2. For any $u, v \in V_\phi$, the pair (u, v) is an edge in E_ϕ if the corresponding assignments are consistent and do not falsify any constraints.

Observe that if ϕ is satisfiable then there exists a N-clique in G_ϕ corresponding to a satisfying assignment. In the case that $\text{val}(\phi) \leq 1 - \frac{1}{M}$, we want to upper bound the density of any subgraph of size N.

First, consider any N-subgraph S corresponding to a full assignment to all N variables (so each vertex in the N-subgraph corresponds to a different variable). Between every pair of vertices in S the assignments will be consistent, but since ϕ is unsatisfiable then at least one constraint is falsified by this assignment. This corresponds to at least one edge missing in S.

Now, if we consider any N-subgraph S' that doesn't correspond to a full assignment, then there will be at least two vertices with corresponding assignments onto the same variable, so these vertices will be inconsistent. So, in this case we again get that there is at least one edge missing in S.

In other words, it is NP-hard to distinguish between:

- G_ϕ contains a clique of size N, and
- any N-subgraph in G_ϕ has density at most $1 - \frac{1}{\binom{N}{2}}$.

Selecting $k = N = n/|\Sigma| = \Omega(n)$ completes the proof. □

There are two natural modifications that allow us to prove non-trivial hardness results for $\text{GapD}k\text{S}(\alpha)$.

1. Instead of starting with any 2CSP_Σ instance, we can start with the output of Dinur's Gap Amplification PCP theorem [11].
2. Instead of creating a vertex for every single-variable assignment, we can create a vertex for every ℓ-variable assignment.

In Section 4.2.1 we study the first modification, and in Section 4.3 we complete the proof of Theorem 1.3 by using both of the modifications.

4.2.1 Strengthening with Dinur's PCP Theorem

Proposition 4.1 can be strengthened using Dinur's PCP theorem.[1]

Theorem 4.2 (Dinur's PCP theorem [11])**.** *Let ψ be a 3Sat instance with N' variables and M' clauses. For some constants ϵ, d, there is a polynomial-time reduction that produces a 2CSP_Σ instance ϕ, with at most $M = M' \operatorname{polylog} M'$ constraints and with constant sized Σ, such that*

- *If* $\text{val}(\psi) = 1$ *then* $\text{val}(\phi) = 1$.
- *If* $\text{val}(\psi) < 1$ *then* $\text{val}(\phi) < 1 - \epsilon$.
- *Every variable in ϕ occurs in at most d constraints.*

Combining Theorem 4.2 with the same reduction from Proposition 4.1, we obtain Proposition 4.3.

Proposition 4.3. *The $\text{GapD}k\text{S}(\alpha)$ problem with $k = \Omega(n)$ is NP-hard when $\alpha > 1 - \Omega(1/n)$.*

Proof. Let ψ be a 3Sat instance and let ϕ be a 2CSP_Σ instance on N variables and M constraints from the output of Theorem 4.2. Use the same reduction as from Proposition 4.1 to create G_ϕ on $n = |\Sigma|N$ vertices.

Similar to the proof of Proposition 4.1, if $\text{val}(\phi) = 1$ then there is a N-clique in G_ϕ corresponding to a satisfying assignment.

In the case that $\text{val}(\phi) < 1 - \epsilon$, let S be any N-subgraph of G_ϕ. Let X be the largest subgraph of S corresponding to assignments to distinct variables. Let $Y = S \setminus X$. So $|Y|$ can be viewed as the number of assignments to repeated variables in S. The main idea is

[1] This particular version of Dinur's PCP theorem comes from [9].

that if $|Y|$ is small then there are many edges missing due to falsified constraints, and if $|Y|$ is large then there are many edges missing due to inconsistencies.

To be more specific, we make the two following observations.

1. S is missing $|Y|$ edges due to inconsistencies. This is because for any vertex in Y, there is a vertex in X with an inconsistent assignment.
2. S is also missing at least $\max(\epsilon M - d(N - |X|), 0) = \max(\epsilon M - d|Y|, 0)$ edges. This is because there are ϵM falsified constraints in ϕ by any full assignment, and there are only $N - |X|$ variables not in X, each occurring in at most d constraints. In other words, within the X component of S, there must be at least $\epsilon M - d(N - |X|)$ missing edges corresponding to falsified constraints. We take the max with 0 because it is possible that $\epsilon M - d|Y|$ is less than 0.

Overall, S is missing at least $|Y| + \max(\epsilon M - d|Y|, 0)$ edges. This value will always be $\Omega(M)$ regardless of the value of $|Y|$ since d is a constant. We also know that $M \geq N$ since each variable occurs in at least one constraint. Hence, S is missing $\Omega(N) = \Omega(n)$ edges, corresponding to a density of at most $1 - \Omega(1/n)$.

So, if we could distinguish between the case that G_ϕ contains a N-clique and the case that every N-subgraph of G_ϕ has density at most $1-\Omega(1/n)$, then we could decide 3Sat. Selecting $k = N = \Omega(n)$ completes the proof. □

4.3 Hardness using Manurangsi's Method

In this section we prove Theorem 1.3. The proof follows Manurangsi's proof [27] very closely, with a few modifications to handle the $k = \Omega(n)$ restriction.

4.3.1 Summary of Manurangsi's Proof

As motivated in the warmup, we start with a CSP produced from Dinur's PCP theorem [11]. For this proof we use a translation of Dinur's PCP theorem that outputs a 3Sat instance instead of a 2CSP_Σ instance.[2]

Theorem 4.4 (Dinur's PCP theorem [11])**.** *Let ψ be a* 3Sat *instance with N' variables and M' clauses. For some constants ϵ, d, there is a polynomial-time reduction that produces a* 3Sat *instance ϕ, with at most $M = M' \operatorname{polylog} M'$ clauses such that*

- *If* $\text{val}(\psi) = 1$ *then* $\text{val}(\phi) = 1$.
- *If* $\text{val}(\psi) < 1$ *then* $\text{val}(\phi) < 1 - \epsilon$.

[2]This particular version of Dinur's PCP theorem comes from [27].

- *Every variable in ϕ occurs in at most d clauses.*

Let ϕ be a 3SAT formula with N variables and M clauses from the output of Theorem 4.4, let ℓ be a constant integer (to be selected later), and construct $G_{\phi,\ell} = (V_{N,\ell}, E_{\phi,\ell})$ in the following way.[3]

- $V_{N,\ell}$ is constructed by adding a vertex for every partial assignment to ℓ variables of ϕ.
- Two vertices $u, v \in V_{N,\ell}$ are adjacent if and only if the corresponding partial assignments are consistent and the partial assignments combined do not falsify any clauses in ϕ.

With this construction, the graph $G_{\phi,\ell}$ has $n = 2^\ell \binom{N}{\ell}$ vertices because there are $\binom{N}{\ell}$ ways to pick ℓ variables and 2^ℓ possible assignments to the ℓ variables. Throughout the proof we utilize the same notation to represent a vertex $V_{N,\ell}$ and a partial assignment to ℓ variables as there is a bijection between the two.

Let $k = \binom{N}{\ell} = 2^{-\ell} n$. We first observe that if ϕ is satisfiable then there exists a clique of size k corresponding to a satisfying assignment. The rest of the section involves upper bounding the density of any k-subgraph when $\text{val}(\phi) < 1 - \epsilon$.

Let S be any k-subgraph of $G_{\phi,\ell}$ and let α be the density of S. The goal is to upper bound the density of S when $\text{val}(\phi) < 1-\epsilon$. To upper bound the density of S, we use a relationship, proved by Alon [1], between the density of a graph and the number of homomorphisms from a complete bipartite graph into the graph.[4]

We also use the following notation, where t is an integer.

- Let $K_{t,t}$ be the complete bipartite graph with t vertices per part.
- Let $\mathcal{K}_{t,t}$ be the set of all homomorphisms from a labelled copy of $K_{t,t}$ into $G_{\phi,\ell}$. We note that these homomorphisms could map multiple vertices in $K_{t,t}$ to the same vertex, so long as they are from the same partition of $K_{t,t}$. Mathematically,

$$\mathcal{K}_{t,t} = \{(L, R) \in (V_{N,\ell})^t \times (V_{N,\ell})^t : \forall u \in L, \forall v \in R, u \neq v \wedge (u, v) \in E_{\phi,\ell}\}.$$

- Let the set of homomorphisms from a labelled copy of $K_{t,t}$ into S be $\mathcal{K}^S_{t,t}$.
- Let A_N be the set of all single variable partial assignments to the N variables. We note that $|A_N| = 2N$.
- In order to upper bound $|\mathcal{K}_{t,t}|$, we break $\mathcal{K}_{t,t}$ into smaller subsets denoted $\mathcal{K}_{t,t}(A, B)$. For any $A, B \subseteq A_N$, $\mathcal{K}_{t,t}(A, B)$ is the set of all $(L, R) \in \mathcal{K}_{t,t}$ such that $\bigcup_{u \in L} u = A$ and $\bigcup_{v \in R} v = B$.

[3] We note that in Manurangsi's original proof he selects $\ell = \frac{N}{\text{polylog } N}$.

[4] See Section 2.1 for the definition of homomorphism.

The proof consists of four steps, which can be summarized by Claims 4.5–4.8. We note that the first three steps are very simple to prove, and the majority of this chapter is the proof of Claim 4.8.

Claim 4.5. *For any integer* $t \geq 1$, $|\mathcal{K}^S_{t,t}| \geq \left(\frac{\alpha\left(\binom{N}{\ell}-1\right)}{\binom{N}{\ell}}\right)^{t^2} \binom{N}{\ell}^{2t}$.

Claim 4.6. *For any integer* $t \geq 1$, $|\mathcal{K}^S_{t,t}| \leq |\mathcal{K}_{t,t}|$.

Claim 4.7. *For any integer* $t \geq 1$, $|\mathcal{K}_{t,t}| \leq 2^{4N} \cdot \max_{A,B \subseteq A_N} |\mathcal{K}_{t,t}(A,B)|$.

Claim 4.8. *In the case that* $\mathrm{val}(\phi) < 1-\epsilon$, *there exists a constant* $\lambda > 0$ *(depending on* ϵ *and* d*) such that for any integer* $t \geq 1$ *and any* $A, B \subseteq A_N$, $|\mathcal{K}_{t,t}(A,B)| \leq \left(2^{-\lambda \ell^2/N}\binom{N}{\ell}\right)^{2t}$.

In Section 4.3.2 we complete the proof of Theorem 1.3 using the above claims. In Section 4.3.3 we prove the Claims 4.5–4.7, and in Section 4.3.4 we prove Claim 4.8.

4.3.2 Completing the Proof of Theorem 1.3

Using the claims from above we now complete the proof of Theorem 1.3.

Theorem 1.3. *For any constant* $\delta \in (0,1)$, *there exists a parameter* $k = \Omega(n)$ *and a constant* $c > 0$ *such that* GapDkS(α) *is* NP*-hard when* $\alpha > 1 - c/n^\delta$.

Proof. Let ψ be any 3Sat instance with N' variables and M' clauses. Create ϕ using Theorem 4.4 (Dinur's PCP theorem). Then ϕ is on N variables and at most $M = M' \operatorname{polylog} M'$ clauses such that each variable occurs in at most d clauses.

Let δ be any constant such that $0 < \delta < 1$, and let $\ell = \lceil 3/\delta \rceil$. Construct $G_{\phi,\ell}$ on $n = 2^\ell \binom{N}{\ell}$ vertices as described in Section 4.3.1. Let $k = \binom{N}{\ell}$. Observe that $k = 2^{-\ell} n = \Omega(n)$ since ℓ is a constant.

First observe that if $\mathrm{val}(\phi) = 1$ then $G_{\phi,\ell}$ has a clique of size $k = \binom{N}{\ell}$ corresponding to a full satisfying assignment.

Now consider the case when $\mathrm{val}(\phi) < 1 - \epsilon$. Let S be any k-subgraph of $G_{\phi,\ell}$, and let α be the density of S. Then applying Claims 4.5–4.8, we find that there exists a constant $\lambda > 0$ (depending on ϵ and d) such that for any integer $t \geq 1$,

$$\left(\frac{\alpha\left(\binom{N}{\ell}-1\right)}{\binom{N}{\ell}}\right)^{t^2} \binom{N}{\ell}^{2t} \leq |\mathcal{K}^S_{t,t}| \leq |\mathcal{K}_{t,t}| \leq 2^{4N} \cdot \max_{A,B \in A_N} |\mathcal{K}_{t,t}(A,B)| \leq 2^{4N} \left(2^{-\lambda \ell^2/N}\binom{N}{\ell}\right)^{2t}.$$

Rearranging for α, we find that

$$\alpha \leq \left(\frac{\binom{N}{\ell}}{\binom{N}{\ell} - 1} \right) 2^{4N/t^2 - 2\lambda\ell^2/(Nt)}.$$

We select $t = \frac{4N^2}{\lambda\ell^2}$ so that $4N/t^2$ is equal to $\lambda\ell^2/(Nt)$, and the expression simplifies to

$$\alpha \leq \left(\frac{\binom{N}{\ell}}{\binom{N}{\ell} - 1} \right) 2^{-\lambda^2\ell^4/(4N^3)}.$$

We can bound $\frac{\binom{N}{\ell}}{\binom{N}{\ell}-1} = 1 + \frac{1}{\binom{N}{\ell}-1} \leq 1 + \frac{2}{\binom{N}{\ell}}$ and $2^{-\lambda^2\ell^4/(4N^3)} \leq 1 - \frac{\lambda^2\ell^4}{8N^3}$. Substituting into the expression for α gives

$$\alpha \leq \left(1 + \frac{2}{\binom{N}{\ell}}\right)\left(1 - \frac{\lambda^2\ell^4}{8N^3}\right) \leq 1 + \frac{2}{\binom{N}{\ell}} - \frac{\lambda^2\ell^4}{8N^3}.$$

We now want to show that the $-\frac{\lambda^2\ell^4}{8N^3}$ term dominates the $\frac{2}{\binom{N}{\ell}}$ term. Observe that $\frac{2}{\binom{N}{\ell}} \leq \frac{2\ell^\ell}{N^\ell}$, and since $\ell > 3$ (but still a constant) then $\frac{2\ell^\ell}{N^\ell} = o(1/N^3)$. Hence, for some constant c,

$$\alpha \leq 1 - \frac{c}{N^3}.$$

Finally, since $n = 2^\ell\binom{N}{\ell} \geq \left(\frac{2N}{\ell}\right)^\ell$, then $N \leq \frac{\ell n^{1/\ell}}{2}$. So, $\alpha \leq 1 - \frac{c'}{n^{3/\ell}}$, where $c' = 8c/\ell^3$ is a constant. We finally substitute $\ell = \lceil 3/\delta \rceil \geq 3/\delta$ to find that

$$\alpha \leq 1 - \frac{c'}{n^\delta}.$$

So, if we could distinguish between the case that $G_{\phi,\ell}$ has a k-clique and the case that every k-subgraph of $G_{\phi,\ell}$ has density less than $1 - \frac{c'}{n^\delta}$ in polynomial time, then we could decide 3Sat in polynomial time. Hence this problem is NP-hard. □

4.3.3 Proofs of Claims 4.5–4.7

Claim 4.5 requires a lemma by Alon [1].

Lemma 4.9 (Alon 2002)**.** *For any $s, t \in \mathbb{N}$, any graph G on n vertices with at least γn^2 edges has at least $(2\gamma)^{st} n^{s+t}$ homomorphisms from a labelled $K_{s,t}$ into G.*

We do not prove Alon's lemma here, but we argue why it makes sense intuitively. If G has γn^2 edges, then the fraction of edges out of all possible edges is roughly 2γ. Observe that

if we are given a random set of $2t$ vertices of G, then the probability that those vertices form a copy of $K_{t,t}$ is roughly $(2\gamma)^{t^2}$. This is because there are t^2 edges in $K_{t,t}$ and an edge should exist in G with probability approximately (2γ). Further, there are n^{2t} ways to select $2t$ vertices, with repetition, from G. Overall, there should be roughly $(2\gamma)^{t^2}n^{2t}$ copies of $K_{t,t}$ in G. The complete proof can be found in Alon's paper [1].

We now prove Claim 4.5.

Claim 4.5. *For any integer $t \geq 1$, $|\mathcal{K}^S_{t,t}| \geq \left(\frac{\alpha\binom{\binom{N}{\ell}}{2}-1)}{\binom{N}{\ell}}\right)^{t^2}\binom{N}{\ell}^{2t}$.*

Proof. Recall that S is a $\binom{N}{\ell}$-subgraph with density α. Then S has $\alpha\binom{\binom{N}{\ell}}{2} = \frac{\alpha\binom{\binom{N}{\ell}}{2}-1)}{2\binom{N}{\ell}}\binom{N}{\ell}^2$ edges. Letting $\gamma = \frac{\alpha\binom{\binom{N}{\ell}}{2}-1)}{2\binom{N}{\ell}}$, $s = t$, and applying Lemma 4.9 completes the proof.[5] □

Claim 4.6. *For any integer $t \geq 1$, $|\mathcal{K}^S_{t,t}| \leq |\mathcal{K}_{t,t}|$.*

Proof. This step is straightforward. Since S is a subgraph of $G_{\phi,\ell}$, the number of copies of $K_{t,t}$ in S is at most the number of copies of $K_{t,t}$ in $G_{\phi,\ell}$. □

In general, Claim 4.6 could be a very bad approximation. However, as we show in Section 4.4, the final hardness result is nearly optimal for this specific reduction technique.

Claim 4.7. *For any integer $t \geq 1$, $|\mathcal{K}_{t,t}| \leq 2^{4N} \cdot \max_{A,B\subseteq A_N} |\mathcal{K}_{t,t}(A,B)|$.*

Proof. Each copy of $K_{t,t}$ in $G_{\phi,\ell}$ is contained in exactly one $\mathcal{K}_{t,t}(A,B)$ set. Hence,

$$|\mathcal{K}_{t,t}| = \sum_{A,B\subseteq A_N} |\mathcal{K}_{t,t}(A,B)| \leq 2^{4N} \cdot \max_{A,B\subseteq A_N} |\mathcal{K}_{t,t}(A,B)|$$

because there are 2^{4N} possible ways to select $A, B \subseteq A_N$. □

Similar to the previous step, Claim 4.7 could be another bad approximation. Despite these two potentially bad approximations, the overall hardness result we prove is almost optimal for this reduction technique (see Section 4.4).

4.3.4 Proof of Claim 4.8

Claim 4.8. *In the case that $\text{val}(\phi) < 1 - \epsilon$, there exists a constant $\lambda > 0$ (depending on ϵ and d) such that for any integer $t \geq 1$ and any $A, B \subseteq A_N$, $|\mathcal{K}_{t,t}(A,B)| \leq \left(2^{-\lambda\ell^2/N}\binom{N}{\ell}\right)^{2t}$.*

[5] The application of Alon's lemma in Manurangsi's proof [27] is slightly weaker, bounding $\frac{\alpha\binom{\binom{N}{\ell}}{2}-1)}{\binom{N}{\ell}} \geq \frac{\alpha}{2}$. In a later step of the proof, we require the slightly stronger result of Claim 4.5.

Proof. Fix any $t \in \mathbb{N}$ and $A, B \subseteq A_N$. To get an initial upper bound on $|\mathcal{K}_{t,t}(A,B)|$, we count the number of possible ways we could select t sets of ℓ elements from A and B:

$$|\mathcal{K}_{t,t}(A,B)| \leq \binom{|A|}{\ell}^t \binom{|B|}{\ell}^t = \left(\frac{|A|!}{(|A|-\ell)!\,\ell!}\frac{|B|!}{(|B|-\ell)!\,\ell!}\right)^t = \left(\frac{\prod_{i=0}^{\ell-1}(|A|-i)(|B|-i)}{\ell!^2}\right)^t.$$

Using the AM-GM inequality (see Section 2.4), $(|A|-i)(|B|-i) \leq \left(\frac{|A|+|B|}{2} - i\right)^2$. Further, we observe that $|A|+|B| \leq 2N$ because, for any variable x and $b \in \{0,1\}$, if the assignment $x = b$ appears in A then $x = \neg b$ cannot appear in B, otherwise there would not be an edge between the corresponding vertices in L and R. So the above bound can be simplified to

$$|\mathcal{K}_{t,t}(A,B)| \leq \left(\frac{\prod_{i=0}^{\ell-1}(N-i)^2}{\ell!^2}\right)^t = \binom{N}{\ell}^{2t}.$$

We can sharpen this bound by considering the probability that each of these selections of $2t$ vertices form a copy of $K_{t,t}$. Recall that A, B are sets of single variable assignments from A_N. We can organize the variables of ϕ into three types based on how each variable appears in A and B.

1. Variable x is of *type* 1 if it occurs at most once in all of the single variable assignments of A and B.
2. Variable x is of *type* 2 if it occurs with both partial assignments in one of A or B. In other words, $(x=0),(x=1) \in A$ or $(x=0),(x=1) \in B$.
3. Variable x is of *type* 3 if it occurs in both A and B with the same assignment. In other words, $(x=b) \in A \cap B$ for some $b \in \{0,1\}$.

Based on the restriction that B cannot contain a contradicting partial assignment to any assignments in A, these are the only three types of variables, and each variable is of exactly one type. See Figure 4.1 for an example.

We now strengthen the $|\mathcal{K}_{t,t}(A,B)| \leq \binom{N}{\ell}^{2t}$ bound by considering three cases based on how many variables there are of each type.

We need the following lemma to complete the proof of Claim 4.8. We prove Lemma 4.10 after finishing this proof.

Lemma 4.10. *Let A be a set of single-variable partial assignments. Let X be a set of pairs of single-variable assignments such that every variable appears in at most one pair. Then the probability that a random selection of ℓ distinct elements of A does not contain both assignments of any pair is at most* $\exp\left(\frac{-\ell^2|X|}{2|A|^2}\right)$.

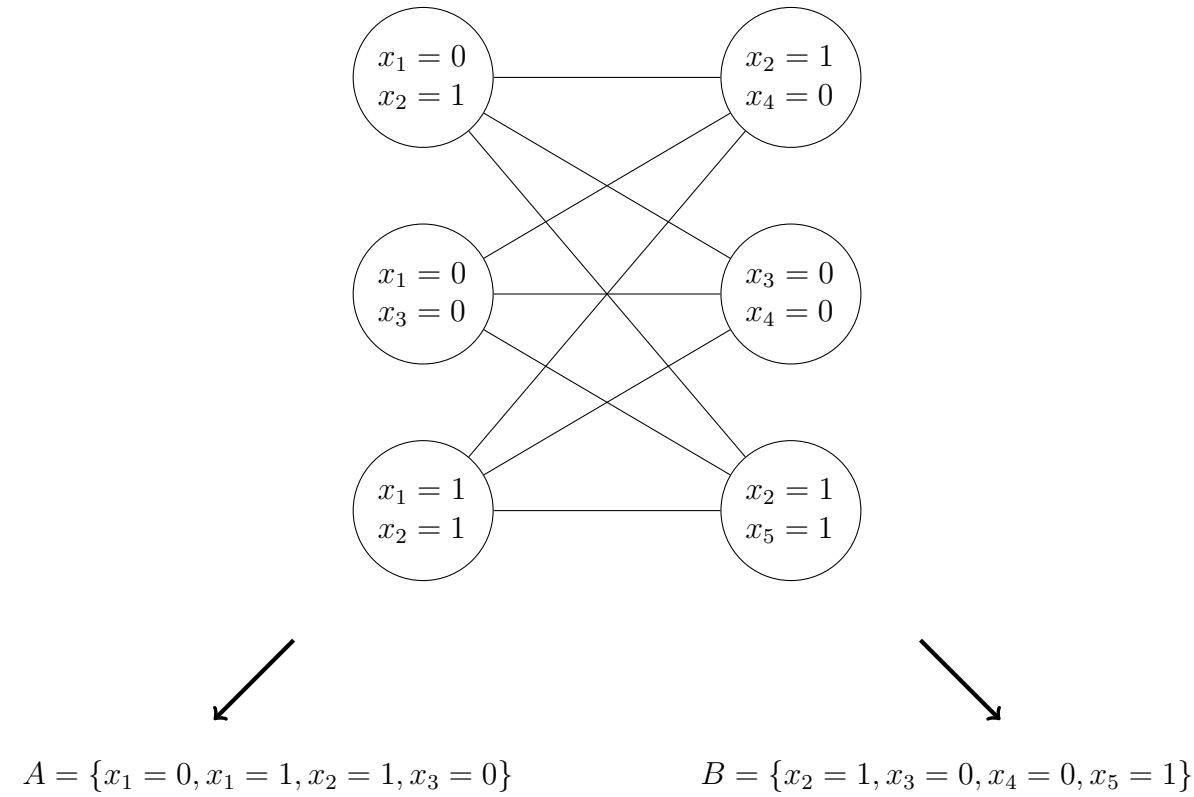

Figure 4.1: Consider the following copy of $K_{3,3}$ which could appear in $G_{\phi,2}$. This copy of $K_{t,t}$ is contained in $\mathcal{K}_{t,t}(A, B)$ for the specific sets A and B shown above. For this subgraph, variables x_4 and x_5 are of type 1 because they appear at most one time in A and B, variable x_1 is of type 2 because it appears with both partial assignments in A, and variables x_2 and x_3 are of type 3 because they have consistent assignments that appear in $A \cap B$.

Let X_1, X_2, X_3 be the sets of variables of the three different types. The main idea is that at least one of $|X_1|, |X_2|, |X_3|$ must be linear in N (the number of variables in ϕ). Let $\beta = \frac{\epsilon}{4d}$ (recall that d is an upper bound on the number of clauses each variable appears in). Consider the following three cases.

Type 1: If $|X_1| \geq \beta N$ then at least βN variables appear at most once in A, B. Hence, $|A| + |B| \leq 2(N - \beta N) + \beta N = 2N(1 - \beta/2)$. Returning to the earlier calculation:

$$|\mathcal{K}_{t,t}(A,B)| \leq \left(\frac{\prod_{i=0}^{\ell-1} \frac{|A|+|B|}{2} - i}{\ell!} \right)^{2t} \leq \left(\frac{\prod_{i=0}^{\ell-1} N(1-\beta/2) - i}{\ell!} \right)^{2t}.$$

Observe that $N(1-\beta/2) - i \leq (1-\beta/2)(N-i) \leq \exp(-\beta/2)(N-i)$. Substituting back into the $|\mathcal{K}_{t,t}(A,B)|$ bound we get

$$|\mathcal{K}_{t,t}(A,B)| \leq \left(\frac{\prod_{i=0}^{\ell-1} \exp(-\beta/2)(N-i)}{\ell!} \right)^{2t} = \left(\exp(-\beta\ell/2) \binom{N}{\ell} \right)^{2t}.$$

Finally, using $\ell < N$ and substituting $\beta = \frac{\epsilon}{4d}$ we get

$$|\mathcal{K}_{t,t}(A,B)| \leq \left(\exp\left(\frac{-\epsilon\ell^2}{8dN} \right) \binom{N}{\ell} \right)^{2t}.$$

Type 2: If $|X_2| \geq \beta N$ then without loss of generality assume that A contains at least $\beta N/2$ variables of type 2. Let X be the set of pairs of assignments to variables in A that are of type 2. A vertex u containing ℓ random variables is legal if no pair of assignments from X are in the vertex. Applying Lemma 4.10 on A and X, we find that the probability that a random selection of ℓ variables is valid is at most $\exp\left(\frac{-\ell^2|X|}{2|A|^2}\right)$. This can be upper bounded by $\exp\left(\frac{-\beta\ell^2}{16N}\right)$ since $|X| \geq \beta N/2$ and $|A| \leq 2N$. Finally, we can use this to simplify the earlier upper bound on $|\mathcal{K}_{t,t}(A,B)|$ by bounding the number of legal vertices that could be selected from A:

$$|\mathcal{K}_{t,t}(A,B)| \leq \left(\exp\left(\frac{-\beta\ell^2}{16N} \right) \binom{|A|}{\ell} \right)^t \binom{|B|}{\ell}^t \leq \left(\exp\left(\frac{-\beta\ell^2}{32N} \right) \binom{N}{\ell} \right)^{2t}.$$

Finally, we substitute $\beta = \frac{\epsilon}{4d}$ to find

$$|\mathcal{K}_{t,t}(A,B)| \leq \left(\exp\left(\frac{-\epsilon\ell^2}{128dN} \right) \binom{N}{\ell} \right)^{2t}.$$

Type 3: If neither of the above cases occur, then $|X_3| \geq (1-2\beta)N$. Let C_3 be the set of clauses from ϕ that only contain variables from X_3. First observe that $|C_3| \geq M - 2\beta dN$ because each of the variables not in X_3 appears in at most d clauses. Now recall that

since $\text{val}(\phi) < 1-\epsilon$, any full assignment falsifies at least ϵM clauses. So any assignment to the variables in X_3 must falsify at least $\epsilon M - 2\beta dN$ clauses from C_3 (otherwise we could augment X_3 with any assignment to the other variables to satisfy more than $M - \epsilon M$ clauses). Simplifying, we find any assignment to the variables in X_3 falsifies at least

$$\epsilon M - 2\beta dN \geq \epsilon N - 2\beta dN = (\epsilon - 2\beta d)N$$

clauses.

Since we select $\beta = \frac{\epsilon}{4d}$, then the above expression is always larger than 0. In particular, any assignment to the variables in X_3 falsifies at least $\epsilon N/2$ clauses.

Using the assignments from $A \cap B$ to the variables in X_3, let the falsified clauses in C_3 be F. Observe that if two variables from a clause in F occur in the same vertex, then that vertex is not part of a valid copy of $K_{t,t}(A, B)$. This is because both A and B contain all the variables from all the clauses in F, hence if two variables from a clause in F occur in the same vertex $v \in L$, then there is no edge between v and the vertex in R containing the third variable of the clause.

Using the clauses from F, we can construct a set X of at least $\frac{\epsilon N}{4d}$ pairs of variables such that each pair of variables share a clause, and no variable appears twice. We can do this by going through F one clause at a time and from each clause add one pair of variables to X, and then remove all clauses from F containing those two variables. The variables from each pair occur in at most $2d$ clauses, which means we can find at least $\frac{\epsilon N}{2}\frac{1}{2d}$ pairs. Again use Lemma 4.10 with $|A| \leq 2N$ and $|X| \geq \frac{\epsilon N}{4d}$ to find that the probability that a random selection of ℓ variables is valid is at least

$$\exp\left(\frac{-\ell^2|X|}{2|A|^2}\right) \leq \exp\left(\frac{-\epsilon\ell^2}{32dN}\right).$$

Again we can upper bound the number of copies of $K_{t,t}$ by upper bounding the number of ways we can legally select ℓ assignments from A:

$$|\mathcal{K}_{t,t}(A, B)| \leq \left(\exp\left(\frac{-\epsilon\ell^2}{32dN}\right)\binom{|A|}{\ell}\right)^t \binom{|B|}{\ell}^t \leq \left(\exp\left(\frac{-\epsilon\ell^2}{64dN}\right)\binom{N}{\ell}\right)^{2t}.$$

Finally, if we select $\lambda = \frac{\epsilon \log e}{128d}$ then in any of the three cases we get that

$$|\mathcal{K}_{t,t}(A, B)| \leq \left(2^{-\lambda\ell^2/N}\binom{N}{\ell}\right)^{2t}. \qquad \square$$

4.3.5 Proof of Lemma 4.10

Lemma 4.10. *Let A be a set of single-variable partial assignments. Let X be a set of pairs of single-variable assignments such that every variable appears in at most one pair. Then the probability that a random selection of ℓ distinct elements of A does not contain both assignments of any pair is at most* $\exp\left(\frac{-\ell^2|X|}{2|A|^2}\right)$.

Proof. Let v be a random selection of ℓ distinct elements of A. Let E_i be the event that at most one of the assignments in pair i of X appears in v, where $1 \leq i \leq |X|$. The events $\{E_1, \ldots, E_{|X|}\}$ are negatively correlated because if one pair is not contained in v then the probability that the other pairs are contained in v goes up. Since the events are negatively correlated, we find

$$\Pr\left[\bigwedge_{i=1}^{|X|} E_i\right] \leq \prod_{i=1}^{|X|} \Pr[E_i].$$

Since each event occurs with the same probability, then

$$\Pr\left[\bigwedge_{i=1}^{|X|} E_i\right] \leq (\Pr[E_1])^{|X|}.$$

We now compute $\Pr[E_1]$. Let $((x_1 = b_1), (x_2 = b_2))$ be the first pair of assignments from X. The event E_1 occurs if one of these two assignments do not appear in v. In other words,

$$\Pr[E_i] = \Pr[(x_1 = b_1) \notin v \vee (x_2 = b_2) \notin v] = 1 - \Pr\left[(x_1 = b_1) \in v \wedge (x_2 = b_2) \in v\right].$$

Observe that $\Pr\left[(x_1 = b_1) \in v \wedge (x_2 = b_2) \in v\right] = \frac{\binom{|A|-2}{\ell-2}}{\binom{|A|}{\ell}}$, which is at least $\frac{\ell^2}{2|A|^2}$.

Substituting back into the earlier bound, we find

$$\Pr\left[\bigwedge_{i=1}^{|X|} E_i\right] \leq (\Pr[E_1])^{|X|} \leq \left(1 - \frac{\ell^2}{2|A|^2}\right)^{|X|} \leq \exp\left(\frac{-\ell^2|X|}{2|A|^2}\right). \qquad \square$$

4.4 Limitations of this Reduction Technique

There were a number of upper bounds used in the proof of Theorem 1.3. In particular, Claims 4.6 and 4.7 use upper bounds that are not tight in general. It is natural to ask whether tighter bounds in the argument could give stronger hardness results. In the proposition below we find dense subgraphs of size $\binom{N}{\ell}$ in $G_{\phi,\ell}$ that demonstrate the limitations of the reduction technique.

Proposition 4.11. *Let ϕ be a* 3Sat *formula with N variables and M clauses from the output of Theorem 4.4 (Dinur's PCP Theorem). If $\mathrm{val}(\phi) = 1 - \gamma < 1 - \epsilon$ for some constant γ (where ϵ is from Theorem 4.4), then for any constant ℓ there exists a $\binom{N}{\ell}$-subgraph in $G_{\phi,\ell}$ with density at least $1 - \Theta\left(\frac{1}{n^{1/\ell}}\right)$, where n is the number of vertices in $G_{\phi,\ell}$.*

Proof. Let A be an assignment to all N variables that satisfies $(1-\gamma)M$ clauses of ϕ, and let S be a $\binom{N}{\ell}$-subgraph of $G_{\phi,\ell}$ corresponding to the assignment A. Observe that

- all the vertices in S correspond to consistent assignments, and
- there are γM clauses that are falsified in ϕ by the assignment A, and $\gamma M \leq \gamma dN$ since each variable occurs in at most d clauses.

So the missing edges in S can only be as a result of falsified clauses. For each falsified clause $C_i(x_{i,1}, x_{i,2}, x_{i,3})$ on variables $x_{i,1}, x_{i,2}$ and $x_{i,3}$, we want to upper bound the number of pairs of vertices in S that together contain all three variables $x_{i,1}, x_{i,2}$ and $x_{i,3}$. There are at most $3\binom{N}{\ell-2}\binom{N}{\ell-1}$ ways to select two sets of ℓ variables such that one set contains at least two of $\{x_{i,1}, x_{i,2}, x_{i,3}\}$ and the other set contains at least the third. Similarly, there are at most $\binom{N}{\ell-3}\binom{N}{\ell}$ ways to select two sets of ℓ variables such that $x_{i,1}, x_{i,2}, x_{i,3}$ all appear in one set. These are the only ways to falsify a clause between two vertices in S, so we can upper bound the number of missing edges in S by

$$\gamma dN\left(3\binom{N}{\ell-2}\binom{N}{\ell-1}+\binom{N}{\ell-3}\binom{N}{\ell}\right).$$

Using the bound $\left(\frac{N}{\ell}\right)^{\ell} \leq \binom{N}{\ell} \leq \left(\frac{Ne}{\ell}\right)^{\ell}$ we find that for any constant ℓ and γ, the subgraph S has density at least

$$1-\frac{\gamma dN\left(3\binom{N}{\ell-2}\binom{N}{\ell-1}+\binom{N}{\ell-3}\binom{N}{\ell}\right)}{\binom{N}{\ell}^2}=1-\Theta\left(\frac{1}{N^2}\right)=1-\Theta\left(\frac{1}{n^{2/\ell}}\right). \qquad \square$$

Proposition 4.11 demonstrates that in the case that $\mathrm{val}(\phi) < 1-\epsilon$, there can exist a $\binom{N}{\ell}$-subgraph in $G_{\phi,\ell}$ with density at least $1-\Theta(n^{-\delta})$, where $\delta = 2/\ell$. This suggests that in order to prove stronger hardness results we need to use a different reduction than the reduction used in Chapter 4.

Chapter 5

GapDkS(α) and MaxClique

Aside from DkS, there is a second very common approximation of the Clique problem. On input G, instead of finding a subgraph with maximal density, the output of MaxClique is a clique of maximal size (the size of the maximal clique is called the *clique number*). This problem has been studied extensively. MaxClique hardness results are shown by proving hardness for the decision version of the problem. In particular, it has been shown that for any constant $\epsilon > 0$, it is NP-hard to distinguish between graphs with clique number larger than $n^{1-\epsilon}$ and graphs with clique number less than n^{ϵ} [19, 32]. This has been strengthened to some subconstant ϵ by Khot [22, 24]. The best known algorithms give approximations of roughly $\left(\frac{n}{\log^3 n}\right)$ in polynomial time [13].

DkS and MaxClique appear to be similar problems, but the algorithms and hardness results are not easily transferable. This disconnection has been mentioned a few times [3, 18, 9]. To demonstrate this point, Proposition 5.1 constructs two graphs with the same clique number, but with very different densities.

Proposition 5.1. *There exists two graphs, each on n vertices, with the same clique number of $\sqrt{n}$ but with densities that have an additive difference of at least $1 - \frac{2}{\sqrt{n}}$.*

Proof. Let G_1 and G_2 be two graphs on n vertices defined as follows.

- Let G_1 be the complete $\sqrt{n}$-partite graph, with $\sqrt{n}$ vertices per part.
- Let G_2 be a graph with a clique of size $\sqrt{n}$ and with no other edges.

First observe that the clique number of both G_1 and G_2 is $\sqrt{n}$. For G_2 this is obvious, but for G_1 observe that we can find a clique of size $\sqrt{n}$ in G_1 by selecting one vertex from each part. Further, this is the maximum sized clique because we cannot select more than one vertex per part.

Next consider the densities of these graphs. In G_1 each vertex has $n - \sqrt{n}$ neighbours, and

so the density of G_1 is

$$\frac{n(n-\sqrt{n})/2}{\binom{n}{2}} = \frac{n-\sqrt{n}}{n-1} \geq 1 - \frac{1}{\sqrt{n}}.$$

On the other extreme, G_2 has density

$$\frac{\binom{\sqrt{n}}{2}}{\binom{n}{2}} = \frac{\sqrt{n}(\sqrt{n}-1)}{n(n-1)} \leq \frac{2}{n}.$$

Hence the difference in densities of G_1 and G_2 is at least $1 - \frac{1}{\sqrt{n}} - \frac{2}{n} \geq 1 - \frac{2}{\sqrt{n}}$. □

Despite the above observation, in Section 5.1 we show that hardness results for the decision version of MaxClique translate to weak hardness results for the GapDkS(α) problem with $k = \Omega(n)$. With this approach, stronger hardness results for MaxClique would correspond to stronger hardness results for GapDkS(α). This appears promising, however based on algorithms for approximating MaxClique we show that this method cannot prove hardness results for GapDkS(α) that beat Theorem 1.3.

5.1 GapDkS(α) Hardness from MaxClique Hardness

In this section we prove the following theorem by translating hardness results for the decision version of MaxClique into hardness results for GapDkS(α). We note that this hardness result was also proved in Proposition 4.3 as a warmup to the proof of Theorem 1.3.

Theorem 5.2. *For some constant $c > 0$, GapDkS(α) with $k = \Omega(n)$ and $\alpha > 1 - \frac{c}{n}$ is* NP*-hard.*

As mentioned above, the main hardness of approximation results for MaxClique show that it is NP-hard to distinguish between graphs with a clique of size $n^{1-\epsilon}$ and graphs for which no graph has size larger than n^{ϵ} [19, 22, 24]. Since the YES instances are graphs with cliques of size $o(n)$, these results cannot be used to give hardness for GapDkS(α) when $k = \Omega(n)$. Instead, we first prove a simple MaxClique hardness result that can be used for the $k = \Omega(n)$ case.

Lemma 5.3. *For some constant $\epsilon > 0$, it is* NP*-Hard to distinguish between graphs with a clique number of $n/3$ and graphs with a clique number less than $(1-\epsilon)n/3$, where n is the number of vertices in the graph.*

Proof. For any 3Sat instance ψ, apply Theorem 4.4 (Dinur's PCP theorem) to create a 3Sat instance ϕ such that if $\text{val}(\psi) = 1$ then $\text{val}(\phi) = 1$, and if $\text{val}(\psi) < 1$ then $\text{val}(\phi) < 1 - \epsilon$ for some constant ϵ.

Let N be the number of variables in ϕ, and M be the number of clauses. Now construct a graph G_ϕ as follows.

- Let the clauses of ϕ be labelled $C_1, \ldots, C_M$. For each clause $C_i(x_{i,1}, x_{i,2}, x_{i,3})$ in ϕ, add 3 vertices to G_ϕ where each vertex corresponds to a single-variable assignment to $x_{i,1}, x_{i,2}, x_{i,3}$ that would satisfy the clause. For example, if the clause is $(x_1 \vee x_2 \vee \neg x_3)$, then we add a vertex corresponding to the assignment $x_1 = 1$, a vertex corresponding to the assignment $x_2 = 1$, and a vertex corresponding to the assignment $x_3 = 0$.
- There is an edge between two vertices if the two vertices are from different clauses and the two corresponding single-variable assignments are consistent.

With this reduction, G_ϕ has $n = 3M$ vertices.[1] First observe that if ϕ is satisfiable then there exists a clique of size $M = n/3$ corresponding to a satisfying assignment.

In the case that $\text{val}(\phi) < 1 - \epsilon$, we make the observation that if G_ϕ contained a clique of size $(1-\epsilon)M$, then we could construct an assignment to satisfy at least $(1-\epsilon)M$ clauses. This is because two vertices coming from the same clause cannot be adjacent, and so a clique of size $(1-\epsilon)M$ would correspond to satisfying $(1-\epsilon)M$ clauses with consistent single-variable assignments. This is a contradiction with $\text{val}(\phi) < 1 - \epsilon$. Hence, when $\text{val}(\phi) < 1 - \epsilon$ then the clique number of G_ϕ is less than $(1-\epsilon)M = (1-\epsilon)n/3$.

So if we could distinguish between graphs with a clique number of $n/3$ and graphs with clique number less than $(1-\epsilon)n/3$ in polynomial time, then we could decide 3Sat in polynomial time. □

In order to translate Lemma 5.3 to a GapDkS(α) hardness result, we need Turán's Theorem [31]. Turán's Theorem relates the clique number with the maximum number of edges that can exist in a graph.

Theorem 5.4 (Turán's Theorem)**.** *Let $G = (V, E)$ be a graph on n vertices and let $k \leq n-1$. If G does not have a $(k+1)$-clique then*

$$|E| \leq \frac{n^2(k-1)}{2k}.$$

This theorem tells us that G_1 from Proposition 5.1 is in fact the highest density graph that has a clique number at most $\sqrt{n}$.

We now complete the proof of Theorem 5.2.

Theorem 5.2. *For some constant $c > 0$, GapDkS(α) with $k = \Omega(n)$ and $\alpha > 1 - \frac{c}{n}$ is* NP*-hard.*

Proof. Let G be a graph on n vertices. Lemma 5.3 states that in general it is NP-hard to distinguish between the case that G has a $n/3$-clique and the case that the largest clique in G is less than $(1-\epsilon)n/3$.

[1] If any clause has less than 3 variables, then by adding dummy variables we ensure that every clause has exactly 3 variables.

In the case when the graph has no clique larger than $(1-\epsilon)n/3$, let S be any $\frac{n}{3}$-subgraph of G. We apply Turán's theorem, with $k = (1-\epsilon)n/3$, to find that the number of edges in S is at most

$$\frac{\left(\frac{n}{3}\right)^2\left(\frac{(1-\epsilon)n}{3}-1\right)}{2(1-\epsilon)n/3} = \frac{\left(\frac{n}{3}\right)^2\left(1-\frac{3}{(1-\epsilon)n}\right)}{2}.$$

Then, the density of S is at most

$$\frac{\left(\frac{n}{3}\right)^2\left(1-\frac{3}{(1-\epsilon)n}\right)}{2\binom{n/3}{2}} = \frac{1-\frac{3}{(1-\epsilon)n}}{1-\frac{3}{n}}.$$

The above expression simplifies to $1-\frac{3\epsilon}{(1-\epsilon)(n-3)}$, so by selecting $c = \frac{3\epsilon}{1-\epsilon}$ and we find that the density of S is at most

$$1-\frac{c}{n-3} \leq 1-\frac{c}{n}.$$

Hence, if for any G we could distinguish between the cases:

- G contains a clique of size $n/3$, and
- Every $\frac{n}{3}$-subgraph of G has density at most $1-c/n$, for some constant c,

in polynomial time, then we could solve any problem in NP in polynomial time. □

5.2 Limitations of MaxClique Hardness Results

If it was possible to start with a stronger MaxClique hardness result than Lemma 5.3, then it would be possible to prove stronger hardness results for GapDkS(α) using the above method. There exist very strong hardness results for MaxClique, and so this appears promising. If it is NP-hard to distinguish between graphs with clique number $\Omega(n)$ and graphs with clique number $n^{o(1)}$, then we could significantly strengthen Theorem 5.2 to beat the result from Theorem 1.3.

Unfortunately, the best hardness results [19, 32, 22, 24] do not show this type of hardness result. Further, this type of hardness result is ruled out by a MaxClique approximation algorithm by Boppana and Halldórsson [8].

Theorem 5.5 (Boppana, Halldórsson 1992)**.** *Let G be a graph on n vertices with a clique of size n/c for some constant $c > 1$. There exists a polynomial-time algorithm to find a clique in G of size $\Omega\left(n^{\frac{1}{c-1}}\right)$.*

Theorem 5.5 tells us that, in polynomial time, it is possible to distinguish between the cases that G has a clique of size $\Omega(n)$, and the case that G has no clique larger than $n^{\Omega(1)}$. Hence it is not possible to use MaxClique hardness results directly to get stronger hardness results for GapDkS(α) when $k = \Omega(n)$.

Chapter 6

Discussion and Open Problems

Informally, this thesis proves that GapDkS(α) with $k = \Omega(n)$:

- is NP-hard to solve when $\alpha > 1 - O(n^{-\delta})$ for any constant $\delta > 0$,
- can be solved in randomized polynomial time when $\alpha < 1 - \Omega(1/\log n)$, and
- can be solved in polynomial time when $\alpha < 1 - \epsilon$ for any constant $\epsilon > 0$.

There are two immediate questions for the GapDkS(α) problem with $k = \Omega(n)$.

Question 1: Can we close the gap by proving stronger hardness results or finding better algorithms? Is there an α range where the problem is neither NP-hard nor in P?

Question 2: Can GapDkS(α) be solved in deterministic polynomial time when $\alpha < 1 - \Omega(1/\log n)$? In other words, can we derandomize the algorithm from Theorem 1.2 and still have a polynomial-time algorithm.

For Question 1, Proposition 4.11 suggests that in order to prove stronger hardness results we need to use a different reduction than the reduction used in Chapter 4. However, it is possible the neither hardness results or algorithms can be strengthened. It is known that for general k, assuming ETH, solving GapDkS(α) when α is a constant requires $n^{\Theta(\log n)}$ time, and there are algorithms that match this bound [9, 16]. This suggests that GapDkS(α) with α constant lies somewhere between P and NP-Complete. Perhaps there is an α range when $k = \Omega(n)$ which also requires $n^{\Theta(\log n)}$ time.

Beyond the $k = \Omega(n)$ restriction, there are a few natural extensions of this work.

Question 3: Can the simplified clique testing algorithm from Theorem 1.2 be used to solve GapDkS(α) on general k?

Question 4: Can we use similar techniques to study the GapDkS(β, α) problem when $k = \Omega(n)$?

In general, the complexity of the GapDkS(α) and GapDkS(β, α) problems are still mostly unknown. GapDkS(α) can be solved in $n^{O(\log n)}$ time when α is a constant [16], suggesting that it is unlikely to be NP-hard when α is a constant. We expect that when $\alpha = 1 - \epsilon$ for some $\epsilon = o(1)$ there is a threshold for when the problem becomes NP-hard.

GapDkS(β, α) is conjectured to be NP-hard when β/α is some polynomial [27], but it has not been shown to be NP-hard when β/α is a constant. This leaves a large gap between the best polynomial-time algorithms which solve GapDkS(β, α) when β/α is roughly $n^{1/4}$ [7].

We hope that our continued work on the GapDkS(α) problem when $k = \Omega(n)$ helps to determine the complexity of the more general problems.

References

[1] Noga Alon. Testing subgraphs in large graphs. *Random Structures & Algorithms*, 21(3-4):359–370, 2002.

[2] Noga Alon, Sanjeev Arora, Rajsekar Manokaran, Dana Moshkovitz, and Omri Weinstein. Inapproximability of densest κ-subgraph from average case hardness. *Unpublished manuscript*, 1, 2011.

[3] Sanjeev Arora, David Karger, and Marek Karpinski. Polynomial time approximation schemes for dense instances of NP-hard problems. *Journal of Computer and System Sciences*, 58(1):193–210, 1999.

[4] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.

[5] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998.

[6] Yuichi Asahiro, Kazuo Iwama, Hisao Tamaki, and Takeshi Tokuyama. Greedily finding a dense subgraph. *Journal of Algorithms*, 34(2):203–221, 2000.

[7] Aditya Bhaskara, Moses Charikar, Eden Chlamtac, Uriel Feige, and Aravindan Vijayaraghavan. Detecting high log-densities: an $O(n^{1/4})$ approximation for densest k-subgraph. In *Proceedings of the 42nd ACM Symposium on Theory of Computing*, pages 201–210, 2010.

[8] Ravi Boppana and Magnús M Halldórsson. Approximating maximum independent sets by excluding subgraphs. *BIT Numerical Mathematics*, 32(2):180–196, 1992.

[9] Mark Braverman, Young Kun Ko, Aviad Rubinstein, and Omri Weinstein. ETH hardness for densest-k-subgraph with perfect completeness. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1326–1341. SIAM, 2017.

[10] Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[11] Irit Dinur. The PCP theorem by gap amplification. *Journal of the ACM*, 54(3):12–es, 2007.

[12] Uriel Feige. Relations between average case complexity and approximation complexity. In *Proceedings of the 34th annual ACM Symposium on Theory of Computing*, pages 534–543, 2002.

[13] Uriel Feige. Approximating maximum clique by removing subgraphs. *SIAM Journal on Discrete Mathematics*, 18(2):219–225, 2004.

[14] Uriel Feige and Michael Langberg. Approximation algorithms for maximization problems arising in graph partitioning. *Journal of Algorithms*, 41(2):174–211, 2001.

[15] Uriel Feige, David Peleg, and Guy Kortsarz. The dense k-subgraph problem. *Algorithmica*, 29(3):410–421, 2001.

[16] Uriel Feige, Michael Seltser, et al. *On the densest k-subgraph problem*. Weizmann Science Press of Israel, 1997.

[17] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.

[18] Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, 1998.

[19] Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182(1):105–142, 1999.

[20] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

[21] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[22] Subhash Khot. Improved inapproximability results for maxclique, chromatic number and approximate graph coloring. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 600–609. IEEE, 2001.

[23] Subhash Khot. Ruling out PTAS for graph min-bisection, dense k-subgraph, and bipartite clique. *SIAM Journal on Computing*, 36(4):1025–1071, 2006.

[24] Subhash Khot and Ashok Kumar Ponnuswami. Better inapproximability results for maxclique, chromatic number and min-3lin-deletion. In *International Colloquium on Automata, Languages, and Programming*, pages 226–237. Springer, 2006.

[25] Guy Kortsarz and David Peleg. On choosing a dense subgraph. In *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science*. IEEE, 1993.

[26] Leonid Anatolevich Levin. Universal sequential search problems. *Problemy peredachi informatsii*, 9(3):115–116, 1973.

[27] Pasin Manurangsi. Almost-polynomial ratio ETH-hardness of approximating densest k-subgraph. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–961, 2017.

[28] Pasin Manurangsi and Dana Moshkovitz. Approximating dense max 2-CSPs. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, page 396, 2015.

[29] Anand Srivastav and Katja Wolf. Finding dense subgraphs with semidefinite programming. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 181–191. Springer, 1998.

[30] Akiko Suzuki and Takeshi Tokuyama. Dense subgraph problems with output-density conditions. *ACM Transactions on Algorithms*, 4(4):1–18, 2008.

[31] Paul Turán. On an external problem in graph theory. *Mat. Fiz. Lapok*, 48:436–452, 1941.

[32] David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 681–690, 2006.