A New Conjecture on Hardness of 2-CSP's with Implications to Hardness of Densest k-Subgraph and Other Problems

Julia Chuzhoy
Toyota Technological Institute at Chicago, USA. Email: `cjulia@ttic.edu`.

Mina Dalirrooyfard
Massachusetts Institute of Technology, USA. Email: `minad@mit.edu`.

Vadim Grinberg
Weizmann Institute of Science, Israel. Email: `vadim.grinberg@weizmann.ac.il`.

Zihan Tan
DIMACS, Rutgers University, USA. Email: `zihantan1993@gmail.con`.

Abstract

We propose a new conjecture on hardness of 2-CSP's, and show that new hardness of approximation results for Densest k-Subgraph and several other problems, including a graph partitioning problem, and a variation of the Graph Crossing Number problem, follow from this conjecture. The conjecture can be viewed as occupying a middle ground between the d-to-1 conjecture, and hardness results for 2-CSP's that can be obtained via standard techniques, such as Parallel Repetition combined with standard 2-prover protocols for the 3SAT problem. We hope that this work will motivate further exploration of hardness of 2-CSP's in the regimes arising from the conjecture. We believe that a positive resolution of the conjecture will provide a good starting point for other hardness of approximation proofs.

Another contribution of our work is proving that the problems that we consider are roughly equivalent from the approximation perspective. Some of these problems arose in previous work, from which it appeared that they may be related to each other. We formalize this relationship in this work.

2012 ACM Subject Classification Mathematics of computing → Approximation algorithms

Keywords and phrases Hardness of Approximation, Densest k-Subgraph

Digital Object Identifier 10.4230/LIPIcs.ITCS.2023.17

Related Version *Full Version*: `https://arxiv.org/abs/2211.05906`

Funding *Julia Chuzhoy*: Supported in part by NSF grant CCF-2006464.
Mina Dalirrooyfard: Part of the work done at Toyota Technological Institute at Chicago.
Vadim Grinberg: Part of the work done at Toyota Technological Institute at Chicago. Supported in part by NSF grant CCF-2006464.
Zihan Tan: Supported by a grant to DIMACS from the Simons Foundation (820931) and NSF grant CCF-2006464.

1 Introduction

In this paper we consider several graph optimization problems, the most prominent and extensively studied of which is Densest k-Subgraph. One of the main motivations of this work is to advance our understanding of the approximability of these problems. Towards this goal, we propose a new conjecture on the hardness of a class of 2-CSP problems, and we show that new hardness of approximation results for all these problems follow from this conjecture. We believe that the conjecture is interesting in its own right, as it can be seen as occupying a middle ground between the d-to-1 conjecture, and the type of hardness of approximation results that one can obtain for 2-CSP problems via standard methods (such as using constant-factor hardness of approximation results for 3-SAT, combined with standard 2-prover protocols and Parallel Repetition). While our conditional hardness of approximation proofs are combinatorial and algorithmic in nature, we hope that this work will inspire complexity theorists to study the conjecture, and also lead to other hardness of approximation proofs that combine both combinatorial and algebraic techniques.

We prove a new conditional hardness of approximation result for Densest k-Subgraph based on our conjecture. In addition to the Densest k-Subgraph problem, we study three other problems. The first problem, called (r,h)-Graph Partitioning, recently arose in the hardness of approximation proof of the Node-Disjoint Paths problem of [18], who mention that the problem appears similar to Densest k-Subgraph, but could not formalize this intuition. We also study a new problem that we call Dense k-Coloring, that can be viewed as a natural middle ground between Densest k-Subgraph and (r,h)-Graph Partitioning. The fourth problem that we study is a variation of the notoriously difficult Minimum Crossing Number problem, that we call Maximum Bounded-Crossing Subgraph. This problem also arose implicitly in [18]. We show that all four problems are roughly equivalent from the approximation perspective, in the regime where the approximation factors are somewhat large (but some of our reductions require quasi-polynomial time). We then derive conditional hardness of approximation results for all these problems based on these reductions and the conditional hardness of Densest k-Subgraph.

The main contribution of this paper is thus twofold: first, we propose a new conjecture on hardness of CSP's and show that a number of interesting hardness of approximation results follow from it. Second, we establish a close connection between the four problems that we study. The remainder of the Introduction is organized as follows. We start by providing a brief overview of the four problems that we study in this paper. We then state our conjecture on hardness of CSP's and put it into context with existing results and well-known conjectures. Finally, we provide a more detailed overview of our results and techniques.

Densest k-Subgraph.

In the Densest k-Subgraph problem, given an n-vertex graph G and an integer $k > 1$, the goal is to compute a subset S of k vertices of G, while maximizing the number of edges in $G[S]$. Densest k-Subgraph is one of the most basic graph optimization problems that has been studied extensively (see e.g. [2, 7–10, 12, 15, 25–30, 36, 41, 44, 46, 47, 52]). At the same time it seems notoriously difficult, and despite this extensive work, our understanding of its approximability is still incomplete. The best current approximation algorithm for Densest k-Subgraph, due to [8], achieves, for every $\varepsilon > 0$, an $O(n^{1/4+\varepsilon})$-approximation, in time $n^{O(1/\varepsilon)}$. Even though the problem appears to be very hard, its hardness of approximation proof has been elusive. For example, no constant-factor hardness of approximation proofs for Densest k-Subgraph are currently known under the standard $\mathsf{P} \neq \mathsf{NP}$ assumption, or even the stronger assumption that $\mathsf{NP} \not\subseteq \mathsf{BPTIME}(n^{\mathrm{poly}\log n})$. In a breakthrough result, Khot [36] proved a factor-c hardness of approximation for Densest k-Subgraph, for some small constant c, assuming that $\mathsf{NP} \not\subseteq \cap_{\varepsilon>0}\mathsf{BPTIME}(2^{n^{\varepsilon}})$. Several other papers proved constant and super-constant hardness of approximation results for Densest k-Subgraph under *average-case* complexity assumptions: namely that no efficient algorithm can refute random 3-SAT or random k-AND formulas [2, 25]. Additionally, a factor $2^{\Omega(\log^{2/3} n)}$-hardness of approximation was shown under assumptions on solving Planted Clique [2]. In a recent breakthrough, Manurangsi [46] proved that, under the Exponential Time Hypothesis (ETH), the Densest k-Subgraph problem is hard to approximate to within factor $n^{1/(\log\log n)^c}$, for some constant c. Proving a super-constant hardness of Densest k-Subgraph under weaker complexity assumptions remains a tantalizing open question that we attempt to address in this paper. Unfortunately, it seems unlikely that the techniques of [46] can yield such a result. In this paper we show that, assuming the conjecture on hardness of 2-CSP that we introduce, Densest k-Subgraph is NP-hard to approximate to within factor $2^{(\log n)^\varepsilon}$, for some constant $\varepsilon > 0$.

The (r, h)-Graph Partitioning Problem.

A recent paper [18] on the hardness of approximation of the Node-Disjoint Paths (NDP) problem formulated and studied a new graph partitioning problem, called (r,h)-Graph Partitioning. The input to the problem is a graph G, and two integers, r and h. The goal is to compute r vertex-disjoint subgraphs $H_1, \ldots, H_r$ of G, such that for each $1 \le i \le r$, $|E(H_i)| \le h$, while maximizing $\sum_{i=1}^{r} |E(H_i)|$. A convenient intuitive way of thinking about this problem is that we are interested in obtaining a balanced partition of the graph G into r vertex-disjoint subgraphs, so that the subgraphs contain sufficiently many edges. Unlike standard graph partitioning problems, that typically aim to minimize the number of edges connecting the different subgraphs in the solution, our goal is to maximize the total number of edges that are contained in the subgraphs. In order to avoid trivial solutions, in which one of the subgraphs contains almost the entire graph G, and the remaining subgraphs are almost empty, we place an upper bound h on the number of edges that each subgraph may contribute towards the solution. Note that the subgraphs H_i of G in the solution need not be vertex-induced subgraphs.

The work of [18] attempted to use (r,h)-Graph Partitioning as a proxy problem for proving hardness of approximation of NDP. Their results imply that NDP is at least as hard to approximate as (r,h)-Graph Partitioning, to within polylogarithmic factors. In order to prove hardness of NDP, it would then be sufficient to show that (r,h)-Graph Partitioning is hard to approximate. Unfortunately, [18] were unable to do so. Instead, they considered a generalization of (r,h)-Graph Partitioning, called (r,h)-Graph Partitioning with Bundles. They showed that NDP is at least as hard as (r,h)-Graph Partitioning with Bundles, and then proved hardness of this new problem. In the (r,h)-Graph Partitioning with Bundles problem, the input is the same as in (r,h)-Graph Partitioning, but now graph G must be bipartite, and, for every vertex v, we are given a partition $\mathcal{B}(v)$ of the set of edges incident to v into subsets that are called *bundles*. We require that, in a solution $(H_1, \ldots, H_r)$ to the problem, for every vertex $v \in V(G)$, and every bundle $\beta \in \mathcal{B}(v)$, at most one edge of β contributes to the solution; in other words, at most one edge of β may lie in $\bigcup_i E(H_i)$. This is a somewhat artificial problem, but this definition allows one to bypass some of the barriers that arise when trying to prove hardness of (r,h)-Graph Partitioning from existing hardness results for CSP's.

It was noted in [18] that the (r,h)-Graph Partitioning problem resembles the Densest k-Subgraph problem for two reasons. First, in Densest k-Subgraph, the goal is to compute a dense subgraph of a given graph, with a prescribed number of vertices. One can think of (r,h)-Graph Partitioning as the problem of computing many vertex-disjoint dense subgraphs of a given graph. Second, natural hardness of approximation proofs for both problems seem to run into the same barriers. It is therefore natural to ask: (i) Can we prove that the (r,h)-Graph Partitioning problem itself is hard to approximate? In particular, can the techniques of [18] be exploited in order to obtain such a proof? and (ii) Can we formalize this intuitive connection between (r,h)-Graph Partitioning and Densest k-Subgraph? In this paper we make progress on both these questions. Our conditional hardness result for

Densest k-Subgraph indeed builds on the ideas from [18] for proving hardness of (r,h)-Graph Partitioning with Bundles. We also provide "almost" approximation-preserving reductions between (r,h)-Graph Partitioning and Densest k-Subgraph: we show that, if there is an efficient factor $\alpha(n)$-approximation algorithm for Densest k-Subgraph, then there is a randomized efficient factor $O(\alpha(n^2) \cdot \mathrm{poly}\log n)$-approximation algorithm to (r,h)-Graph Partitioning. We also provide a reduction in the opposite direction: we prove that, if there is an efficient $\alpha(n)$-approximation algorithm for (r,h)-Graph Partitioning, then there is a randomized algorithm for Densest k-Subgraph, that achieves approximation factor $O\left((\alpha(n^{O(\log n)}))^3 \cdot \log^2 n\right)$, in time $n^{O(\log n)}$. Therefore, we prove that Densest k-Subgraph and (r,h)-Graph Partitioning are roughly equivalent from the approximation perspective (at least for large approximation factors and quasi-polynomial running times). Combined with our conditional hardness of approximation for Densest k-Subgraph, our results show that, assuming the conjecture on hardness of 2-CSP that we introduce, for some constant $0 < \varepsilon \le 1/2$, there is no efficient $2^{(\log n)^{\varepsilon}}$-approximation algorithm for (r,h)-Graph Partitioning, unless $\mathsf{NP} \subseteq \mathsf{BPTIME}(n^{O(\log n)})$.

Maximum Bounded-Crossing Subgraph.

The third problem that we study is a variation of the classical Minimum Crossing Number problem. In the Minimum Crossing Number problem, given an input n-vertex graph G, the goal is to compute a drawing of G in the plane while minimizing the number of crossings in the drawing. We define the notions of graph drawing and crossings formally in the Preliminaries, but these notions are quite intuitive and the specifics of the definition are not important in this high-level overview.

The Minimum Crossing Number problem was initially introduced by Turán [54] in 1944, and has been extensively studied since then (see, e.g., [13,14,16,19,20,32,33], and also [48–51] for excellent surveys). But despite all this work, most aspects of the problem are still poorly understood. A long line of work [16, 17, 20, 21, 24, 32, 33, 43] has recently led to the first sub-polynomial approximation algorithm for the problem in *low degree graphs*. Specifically, [21] obtain a factor $O\left(2^{O((\log n)^{7/8} \log\log n)} \cdot \Delta^{O(1)}\right)$-approximation algorithm for Minimum Crossing Number, where Δ is the maximum vertex degree. To the best of our knowledge, no non-trivial approximation algorithms are known for the problem when vertex degrees in the input graph G can be arbitrary. However, on the negative side, only APX-hardness is known for the problem [3,11]. As the current understanding of the Minimum Crossing Number problem from the approximation perspective is extremely poor, it is natural to study hardness of approximation of its variants.

Let us consider two extreme variations of the Minimum Crossing Number problem. The first variant is the Minimum Crossing Number problem itself, where we need to draw an input graph G in the plane with fewest crossings. The second variant is where we need to compute a subgraph G' of the input graph G that is planar, while maximizing $|E(G')|$. The latter problem has a simple constant-factor approximation algorithm, obtained by letting G' be any spanning forest of G (this is since a planar n-vertex graph may only have $O(n)$ edges).

In this paper we study a variation of the Minimum Crossing Number problem, that we call Maximum Bounded-Crossing Subgraph, which can be viewed as an intermediate problem between these two extremes. In the Maximum Bounded-Crossing Subgraph problem, given an n-vertex graph G and an integer $L > 0$, the goal is to compute a subgraph $H \subseteq G$, such that H has a plane drawing with at most L crossings, while maximizing $|E(H)|$. Unless we are interested in constant approximation factors, this problem is only interesting when the bound L on the number of crossings is $\Omega(n)$. This is since, from the Crossing Number Inequality [1, 42], if $|E(G)| \ge 4|V(G)|$, then the crossing number of G is at least

$\Omega(|E(G)|^3/|V(G)|^2)$. Therefore, for $L = O(n)$, a spanning tree provides a constant-factor approximation to the problem. We emphasize that the focus here is on dense graphs, whose crossing number may be as large as $\Omega(n^4)$.

The Maximum Bounded-Crossing Subgraph problem was implicitly used in [18] for proving hardness of approximation of NDP, as an intermediate problem, in the reduction from (r,h)-Graph Partitioning with Bundles to NDP. Their work suggests that there may be a connection between (r,h)-Graph Partitioning and Maximum Bounded-Crossing Subgraph, even though the two problems appear quite different. In this paper we prove that the two problems are roughly equivalent from the approximation perspective: if there is an efficient factor $\alpha(n)$-approximation algorithm for (r,h)-Graph Partitioning, then there is an efficient $O(\alpha(n) \cdot \text{poly}\log n)$-approximation algorithm for Maximum Bounded-Crossing Subgraph. On the other hand, an efficient $\alpha(n)$-approximation algorithm for Maximum Bounded-Crossing Subgraph implies an efficient $O((\alpha(n))^2 \cdot \text{poly}\log n)$-approximation algorithm for (r,h)-Graph Partitioning. Combined with our conditional hardness of approximation for (r,h)-Graph Partitioning, we get that, assuming the conjecture on hardness of 2-CSP that we introduce, for some constant $0 < \varepsilon \leq 1/2$ there is no efficient $2^{(\log n)^\varepsilon}$-approximation algorithm for Maximum Bounded-Crossing Subgraph, unless $\mathsf{NP} \subseteq \mathsf{BPTIME}(n^{O(\log n)})$.

Dense k-Coloring.

The fourth and last problem that we consider is Dense k-Coloring. In this problem, the input is an n-vertex graph G and an integer k, such that n is an integral multiple of k. The goal is to partition $V(G)$ into n/k disjoint subsets $S_1, \ldots, S_{n/k}$, of cardinality k each, so as to maximize $\sum_{i=1}^{n/k} |E(S_i)|$. This problem can be viewed as an intermediate problem between Densest k-Subgraph and (r,h)-Graph Partitioning. The connection to (r,h)-Graph Partitioning seems clear: in both problems, the goal is to compute a large collection of disjoint subgraphs of the input graph G, that contain many edges of G. While in (r,h)-Graph Partitioning we place a limit on the number of edges in each subgraph, in Dense k-Coloring we require that each subgraph contains exactly k vertices. The connection to the Densest k-Subgraph problem is also clear: while in Densest k-Subgraph the goal is to compute a single dense subgraph of G containing k vertices, in Dense k-Coloring we need to partition G into many dense subgraphs, containing k vertices each. We show reductions between the Dense k-Coloring and the Densest k-Subgraph problem in both directions, that provide very similar guarantees to the reductions between (r,h)-Graph Partitioning and Densest k-Subgraph. In particular, our results show that, assuming the conjecture on the hardness of 2-CSP that we introduce, for some constant $0 < \varepsilon \leq 1/2$, there is no efficient $2^{(\log n)^\varepsilon}$-approximation algorithm for Dense k-Coloring, unless $\mathsf{NP} \subseteq \mathsf{BPTIME}(n^{O(\log n)})$.

Our Conjecture on Hardness of 2-CSP's

We now turn to describe our new conjecture on hardness of 2-CSP's. We consider the following bipartite version of the Constraint Satisfaction Problem with 2 variables per constraint (2-CSP). The input consists of two sets X and Y of variables, together with an integer $A \geq 1$. Every variable in $X \cup Y$ takes values in $[A] = \{1, \ldots, A\}$. We are also given a collection $\mathcal{C}$ of constraints, where each constraint $C(x, y) \in \mathcal{C}$ is defined over a pair of variables $x \in X$ and $y \in Y$. For each such constraint, we are given a truth table that, for every pair of assignments a to x and a' to y, indicates whether (a, a') *satisfy* the constraint. The *value* of the CSP is the largest fraction of constraints that can be simultaneously satisfied by an assignment to the variables. For given values $0 < s < c \leq 1$, the (c, s)-Gap-CSP problem is the problem of distinguishing CSP's of value at least c from those of value at most s.

We can associate, to each constraint $C = C(x, y) \in \mathcal{C}$, a bipartite graph $G_C = (L, R, E)$, where $L = R = [A]$, and there is an edge (a, a') in E iff the assignments a to x and a' to y satisfy C. Notice that instance $\mathcal{I}$ of the Bipartite 2-CSP problem is completely defined by $X, Y, A, \mathcal{C}$, and the graphs in $\{G_C\}_{C\in\mathcal{C}}$, so we will denote $\mathcal{I} = (X, Y, A, \mathcal{C}, \{G_C\}_{C\in\mathcal{C}})$. We let the *size* of instance $\mathcal{I}$ be $\text{size}(\mathcal{I}) = |\mathcal{C}| \cdot A^2 + |X| + |Y|$. We sometimes refer to A as the *size of the alphabet for instance* $\mathcal{I}$. We say that instance $\mathcal{I}$ of 2-CSP is *d-to-d'* iff for every constraint C, every vertex of G_C that lies in L has degree at most d, and every vertex that lies in R has degree at most d'. (We note that this is somewhat different from the standard definition, that requires that all vertices in L have degree exactly d and all vertices of R have degree exactly d'. In the standard definition, the alphabet sizes for variables in X and Y may be different, that is, variables in X take values in $[A]$ and variables of Y take values in $[A']$ for some integers A, A'. However, this difference is insignificant to our discussion, and it is more convenient for us to use this slight variation of the standard definition).

The famous Unique-Games Conjecture of Khot [35] applies to 1-to-1 CSP's. The conjecture states that, for any $0 < \varepsilon < 1$, there is a large enough value A, such that the $(1-\varepsilon, \varepsilon)$-Gap-CSP problem is NP-hard for 1-to-1 instances with alphabet size A. The conjecture currently remains open, though interesting progress has been made on the algorithmic side: the results of [4] provide an algorithm for the problem with running time $2^{n^{O(1/\varepsilon^{1/3})}}$.

A conjecture that is closely related to the Unique-Games Conjecture is the d-to-1 Conjecture of Khot [35]. The conjecture states that, for every $0 < \varepsilon < 1$, and $d > 0$, there is a large enough value A, such that the $(1, \varepsilon)$-Gap-CSP problem in d-to-1 instances with alphabet size A is NP-hard.

Håstad [31] proved the following nearly optimal hardness of approximation results for CSP's: he showed that for every $0 < \varepsilon < 1$, there are values d and A, such that the problem of $(1, \varepsilon)$-Gap-CSP in d-to-1 instances with alphabet size A is NP-hard. The value d, however, depends exponentially on $\text{poly}(1/\varepsilon)$ in this result. In contrast, in the d-to-1 Conjecture, both d and ε are fixed, and d may not have such a strong dependence on $1/\varepsilon$.

On the algorithmic side, the results of [4, 53] provide an algorithm for (c, s)-Gap-CSP on d-to-1 instances. The running time of the algorithm is $2^{n^{O(1/(\log(1/s))^{1/2})}}$, where the $O(\cdot)$ notation hides factors that are polynomial in d and A.

A recent breakthrough in this area is the proof of the 2-to-2 conjecture (now theorem), that builds on a long sequence of work [5, 6, 22, 23, 37–40]. The theorem proves that for every $0 < \varepsilon < 1$, there is a large enough value A, such that the $(1 - \varepsilon, \varepsilon)$-Gap-CSP problem is NP-hard on 2-to-2 instances with alphabet size A.

In this paper, we propose the following conjecture regarding the hardness of Gap-CSP in d-to-d instances.

▶ **Conjecture 1.** *There is a constant $0 < \varepsilon \leq 1/2$, such that it is* NP*-hard to distinguish between $d(n)$-to-$d(n)$ instances of 2-*CSP *of size n, that have value at least $1/2$, and those of value at most $s(n)$, where $d(n) = 2^{(\log n)^{\varepsilon}}$ and $s(n) = 1/2^{64(\log n)^{1/2+\varepsilon}}$.*

We now compare this conjecture to existing conjectures and results in this area that we are aware of. First, in contrast to the d-to-1 conjecture, we allow the parameter d and the soundness parameter s to be functions of n – the size of the input instance. Note that the size of the input instance depends on the alphabet size A, so, unlike in the setting of the d-to-1 conjecture, A may no longer be arbitrarily large compared to d and s.

The hardness of approximation result of Håstad [31] for d-to-d CSP's only holds when d depends exponentially on $\text{poly}(1/s)$, (in particular it may not extend to the setting where

$s(n) = 1/2^{64(\log n)^{1/2+\varepsilon}}$, since the size n of the instance depends polynomially on $d(n)$).

We can also combine standard constant hardness of approximation results for CSP's (such as, for example, 3-SAT) with the Parallel Repetition theorem, to obtain NP-hardness of $(1, s(n))$-Gap-CSP on $d(n)$-to-$d(n)$ instances. Using this approach, if we start from an instance of CSP of size N and a constant hardness gap (with perfect completeness), after ℓ rounds of parallel repetition, we obtain hardness of $(1, s)$-Gap-CSP on d-to-d instances with $s = 2^{-O(\ell)}$, $d = 2^{O(\ell)}$, and the resulting instance size $n = N^{O(\ell)}$. Note that $d = (1/s)^{\Theta(1)}$ holds, wich is different from the relationship between these parameters required by the conjecture. Specifically, by setting the number of repetition to be $\ell = \Theta\left((\log N)^{(1/2+\varepsilon)/(1/2-\varepsilon)}\right)$, we can ensure the desired bound $s(n) = 1/2^{64(\log n)^{1/2+\varepsilon}}$. However, in this setting, we also get that $d(n) = 2^{\Omega((\log n)^{1/2+\varepsilon})}$, which is significantly higher than the desired value $d(n) = 2^{(\log n)^{\varepsilon}}$.

Lastly, one could attempt to combine the recent proof of the 2-to-2 conjecture with Parallel Repetition in order to reap the benefits of both approaches, but the resulting parameters also fall short of the ones stated in the conjecture.

From the above discussion, one can view Conjecture 1 as occupying a middle ground between the d-to-1 conjecture, and the results one can obtain via standard techniques of amplifying a constant hardness of a CSP, such as 3SAT, via Parallel Repetition. We note that, while the conjecture appears closely related to the Unique Games Conjecture and d-to-1 conjecture, we are not aware of any additional formal connections, except for those mentioned above.

We now proceed to discuss our results and techniques in more detail.

1.1 A More Detailed Overview of our Results and Techniques

In addition to posing Conjecture 1 that we already described above, we prove conditional hardness of approximation of the four problems that we consider. We also prove that all four problems are roughly equivalent approximation-wise. We now discuss the conditional hardness of approximation for Densest k-Subgraph and the connections between the four problems that we establish.

Conditional Hardness of Densest k-Subgraph.

Our first result is a conditional hardness of Densest k-Subgraph. Specifically, we prove that, assuming that Conjecture 1 holds and that P $\neq$ NP, for some $0 < \varepsilon \leq 1/2$, there is no efficient approximation algorithm for Densest k-Subgraph problem that achieves approximation factor $2^{(\log N)^{\varepsilon}}$, where N is the number of vertices in the input graph.

We now provide a brief overview of our techniques. The proof of the above result employs a Cook-type reduction, and follows some of the ideas that were introduced in [18]. We assume for contradiction that there is a factor-α algorithm $\mathcal{A}$ for the Densest k-Subgraph problem, where $\alpha = 2^{(\log N)^{\varepsilon}}$. Given an input instance $\mathcal{I}$ of the 2-CSP problem of size n, that is a $d(n)$-to-$d(n)$ instance, we construct a constraint graph H representing $\mathcal{I}$. We gradually decompose graph H into a collection $\mathcal{H}$ of disjoint subgraphs, such that, for each subgraph $H' \in \mathcal{H}$, we can either certify that the value of the corresponding instance of 2-CSP is at most $1/4$, or it is at least β, for some carefully chosen parameter β. In order to compute the decomposition, we start with $\mathcal{H} = \{H\}$. If, for a graph $H' \in \mathcal{H}$, we certified that the corresponding instance of 2-CSP has value at most $1/4$, or at least β, then we say that graph H' is *inactive*. Otherwise, we say that it is *active*. As long as $\mathcal{H}$ contains at least one active graph, we perform iterations. In each iteration, we select an arbitrary active graph $H' \in \mathcal{H}$ to process. In order to process H', we consider an *assignment graph* G' associated with H', that contains a vertex for every variable-assignment pair (x, a), where x is a variable whose corresponding vertex belongs to H'. We view G' as an instance of the Densest k-Subgraph problem, for an appropriately chosen parameter k, and apply the approximation algorithm $\mathcal{A}$ for Densest k-Subgraph to it. Let S be the set of vertices of G' that Algorithm $\mathcal{A}$ computes as a solution to this instance. Note that S is a set of vertices in the assignment graph G', while $\mathcal{H}$ is a family of subgraphs of the constraint graph H. We exploit the set S of vertices in order to either (i) compute a large subset $E' \subseteq E(H')$ of edges, such that, if we denote by $\mathcal{C}' \subseteq \mathcal{C}$ the set of constraints corresponding to E', then at most $1/4$ of the constraints of $\mathcal{C}'$ can be simultaneously satisfied; or (ii) compute a large subset $E' \subseteq E(H')$ of edges as above, and certify that at least a β-fraction of such constraints can be satisfied; or (iii) compute a subgraph $H'' \subseteq H'$, such that $|V(H'')| \ll |V(H')|$, and the number of edges contained in graphs H'' and $H' \setminus V(H'')$ is sufficiently large compared to $E(H')$. In the former two cases, we replace H' with graph $H'[E']$ in $\mathcal{H}$, and graph $H'[E']$ becomes inactive. In the latter case, we replace H' with two graphs: H'' and $H' \setminus V(H'')$, that both remain active. The algorithm terminates once every graph in $\mathcal{H}$ is inactive. The crux of the analysis of the algorithm is to show that, when the algorithm terminates, the total number of edges lying in the subgraphs $H' \in \mathcal{H}$ is high, compared to $|E(H)|$. The specific fraction of edges that remain in the subgraphs $H' \in \mathcal{H}$ is governed by the parameters $s(n)$ and $d(n)$, and the specific relationship between these parameters in Conjecture 1 is selected to ensure that many edges remain in the graphs of $\mathcal{H}$ when the algorithm terminates. The algorithm for decomposing graph H into subgraphs and its analysis employ some of the techniques and ideas introduced in [18], and is very similar in spirit to the hardness of approximation proof of the (r,h)-Graph Partitioning with Bundles problem, though details are different. We employ this decomposition algorithm multiple times, in order to obtain a partition $(E_0, E_1, \ldots, E_z)$ of the set $E(H)$ of edges of the constraint graph into a small number of subsets, such that, among the constraints corresponding to the edges of E_0, at most a $1/4$-fraction can be satisfied by any assignment to $X \cup Y$, and, for all $1 \leq i \leq z$, a large fraction of constraints corresponding to edges of E_i can be satisfied by some assignment. Depending on the cardinality of the set E_0 of edges we then determine whether $\mathcal{I}$ is a Yes-Instance or a No-Instance.

Reductions from Dense k-Coloring and (r,h)-Graph Partitioning to Densest k-Subgraph.

We show that, if there is an efficient factor $\alpha(n)$-approximation algorithm for the Densest k-Subgraph problem, then there is a randomized efficient $O(\alpha(n^2) \cdot \operatorname{poly}\log n)$-approximation algorithm for Dense k-Coloring, and a randomized efficient $O(\alpha(n^2) \cdot \operatorname{poly}\log n)$-approximation algorithm for (r,h)-Graph Partitioning. The two reductions are very similar, so we focus on describing the first one. We believe that the reduction is of independent interest, and uses unusual techniques.

We assume that there is an $\alpha(n)$-approximation algorithm for the Densest k-Subgraph problem. In order to obtain an approximation algorithm for Dense k-Coloring, we start by formulating a natural LP-relaxation for the problem. Unfortunately, this LP-relaxation has a large number of variables: roughly $n^{\Theta(k)}$, where n is the number of vertices in the input graph and k is the parameter of the Dense k-Coloring problem instance. We then show an efficient algorithm, that, given a solution to the LP-relaxation, whose support size is bounded by $\operatorname{poly}(n)$, computes an approximate integral solution to the Dense k-Coloring problem.

The main challenge is that, since the LP relaxation has $n^{\Theta(k)}$ variables, it is unclear how to solve it efficiently. We consider the dual linear program, that has $\operatorname{poly}(n)$ variables and $n^{\Theta(k)}$ constraints. Using the $\alpha(n)$-approximation algorithm for Densest k-Subgraph as a subroutine, we design an approximate separation oracle for the dual LP, that allows us to solve the original LP-relaxation for Dense k-Coloring, obtaining a solution whose support size is bounded by $\text{poly}(n)$. By applying the LP-rounding approximation algorithm to this solution, we obtain the desired approximate solution to the input instance of Dense k-Coloring.

Reductions from Densest k-Subgraph to (r,h)-Graph Partitioning and Dense k-Coloring.

We prove that, if there is an efficient $\alpha(n)$-approximation algorithm for Dense k-Coloring, then there is a randomized algorithm for the Densest k-Subgraph problem, whose running time is $n^{O(\log n)}$, that with high probability obtains an $O(\alpha(n^{O(\log n)}) \cdot \log n)$-approximate solution to the input instance of the problem. We also show a similar reduction from Densest k-Subgraph to (r,h)-Graph Partitioning, but now the resulting approximation factor for Densest k-Subgraph becomes $O((\alpha(n^{O(\log n)}))^3 \cdot \log^2 n)$. By combining these reductions with our conditional hardness result for Densest k-Subgraph, we get that, assuming Conjecture 1, for some constant $0 < \varepsilon \leq 1/2$, there is no efficient $2^{(\log n)^\varepsilon}$-approximation algorithm for (r,h)-Graph Partitioning and for Dense k-Coloring, unless $\mathsf{NP} \subseteq \mathsf{BPTIME}(n^{O(\log n)})$.

The two reductions are very similar; we focus on the reduction to Dense k-Coloring in this overview. Our construction is inspired by the results of [34], and we borrow some of our ideas from them. Assume that there is an efficient $\alpha(n)$-approximation algorithm for Dense k-Coloring. Let G be an instance of the Densest k-Subgraph problem. The main difficulty in the reduction is that it is possible that G only contains one very dense subgraph induced by k vertices, while the Dense k-Coloring problem requires that the input graph G can essentially be partitioned into many such dense subgraphs. To overcome this difficulty, we construct a random "inflated" bipartite graph H, that contains $n^{O(\log n)}$ vertices, where $n = |V(G)|$. Every vertex of G is mapped to some vertex of H at random, while every edge of G is mapped to a large number of edges of H. This allows us to ensure that, if G contains a subgraph G' induced by a set of k vertices, where $|E(G')| = R$, then graph H can essentially be partitioned into a large number of subgraphs that contain k vertices each, and many of them contain close to R edges. Therefore, we can apply our $\alpha(n)$-approximation algorithm for Dense k-Coloring to the new graph H. The main challenge in the reduction is that, while this approximation algorithm is guaranteed to return a large number of disjoint dense subgraphs of H, since every edge of G contributes many copies to H, it is not clear that one can extract a single dense subgraph of G from dense subgraphs of H. The main difficulty in the reduction is to ensure that, on the one hand, a single k-vertex dense subgraph in G can be translated into $|V(H)|/k$ dense subgraphs of H; and, on the other hand, a dense k-vertex subgraph of H can be translated into a dense subgraph of G on k vertices. We build on and expand the ideas from [34] in order to ensure these properties.

Reductions between (r,h)-Graph Partitioning and Maximum Bounded-Crossing Subgraph.

Lastly, we provide reductions between (r,h)-Graph Partitioning and Maximum Bounded-Crossing Subgraph in both directions. First, we show that, if there is an efficient factor $\alpha(n)$-approximation algorithm for (r,h)-Graph Partitioning, then there is an efficient $O(\alpha(n) \cdot \text{poly}\log n)$-approximation algorithm for Maximum Bounded-Crossing Subgraph. On the other hand, an efficient $\alpha(n)$-approximation algorithm for Maximum Bounded-Crossing Subgraph implies an efficient $O((\alpha(n))^2 \cdot \text{poly}\log n)$-approximation algorithm for (r,h)-Graph Partitioning. Combined with our conditional hardness of approximation for (r,h)-Graph Partitioning, we get that, assuming Conjecture 1, for some constant $0 < \varepsilon \leq 1/2$, there is no efficient $2^{(\log n)^\varepsilon}$-approximation algorithm for Maximum Bounded-Crossing Subgraph, unless $\mathsf{NP} \subseteq \mathsf{BPTIME}(n^{O(\log n)})$.

Both these reductions exploit the following connection between crossing number and graph partitioning: if a graph G has a drawing with at most L crossings, then there is a balanced cut in G, containing at most $O\left(\sqrt{L+\Delta\cdot|E(G)|}\right)$ edges, where Δ is maximum vertex degree in G. This result can be viewed as an extension of the classical Planar Separator Theorem of [45]. Another useful fact exploited in both reductions is that any graph G with m edges has a plane drawing with at most m^2 crossings. In particular, if $\mathcal{H}=\{H_1,\dots,H_r\}$ is a solution to an instance of the (r,h)-Graph Partitioning problem on graph G, then there is a drawing of graph $H=\bigcup_{i=1}^r H_i$, in which the number of crossings is bounded by $r\cdot h^2$. These two facts establish a close relationship between the (r,h)-Graph Partitioning and Maximum Bounded-Crossing Subgraph problems, that are exploited in both our reductions.

We have now obtained a chain of reductions, showing that all four problems, Densest k-Subgraph, Dense k-Coloring, (r,h)-Graph Partitioning, and Maximum Bounded-Crossing Subgraph are almost equivalent from approximation viewpoint, if we consider sufficiently large approximation factors and allow randomized quasi-polynomial time algorithms. We also obtain conditional hardness of approximation results for all four problems based on Conjecture 1.

Organization.

We start with preliminaries in Section 2. In Section 3 we provide the conditional hardness of approximation proof for the Densest k-Subgraph problem. In Section 4 we provide our reductions from Dense k-Coloring and (r,h)-Graph Partitioning to Densest k-Subgraph, and in Section 5 we provide reductions in the opposite direction. Lastly, in Section 6 we provide reductions between (r,h)-Graph Partitioning and Maximum Bounded-Crossing Subgraph. Due to lack of space, some of the proofs are deferred to the full version of the paper.

2 Preliminaries

By default, all logarithms are to the base of 2. For a positive integer N, we denote by $[N]=\{1,2,\dots,N\}$. All graphs are finite, simple and undirected. We say that an event holds with high probability if the probability of the event is $1-1/n^c$ for a large enough constant c, where n is the number of vertices in the input graph.

2.1 General Notation

Let G be a graph and let S be a subset of its vertices. We denote by $G[S]$ the subgraph of G induced by S. For two disjoint subsets A,B of vertices of G, we denote by $E_G(A,B)$ the set of all edges with one endpoint in A and the other endpoint in B, and we denote by $E_G(A)$ the set of all edges with both endpoints in A. Given a graph G and a vertex $v\in V(G)$, we denote by $\deg_G(v)$ the degree of v in G. For a subset S of vertices of G, its *volume* is $\mathrm{vol}_G(S)=\sum_{v\in S}\deg_G(v)$. We sometimes omit the subscript G if it is clear from the context.

Given a graph G, a drawing φ of G is an embedding of G into the plane, that maps every vertex v of G to a point (called the *image of* v and denoted by $\varphi(v)$), and every edge e of G to a simple curve (called the *image of* e and denoted by $\varphi(e)$), that connects the images of its endpoints. If e is an edge of G and v is a vertex of G, then the image of e may only contain the image of v if v is an endpoint of e. Furthermore, if some point p belongs to the images of three or more edges of G, then p must be the image of a common endpoint of all edges e with $p\in\varphi(e)$. We say that two edges e,e' of G *cross* at a point p, if $p\in\varphi(e)\cap\varphi(e')$, and p is not the image of a shared endpoint of these edges. Given a graph G and a drawing

φ of G in the plane, we use $\mathsf{cr}(\varphi)$ to denote the number of crossings in φ, and the crossing number of G, denoted by $\mathsf{CrN}(G)$, is the minimum number of crossings in any drawing of G.

2.2 Problem Definitions and Additional Notation

In this paper we consider the following four problems: Densest k-Subgraph, Dense k-Coloring, (r,h)-Graph Partitioning and Maximum Bounded-Crossing Subgraph. We now define the problems, along with some additional notation.

Densest k-Subgraph.

In the Densest k-Subgraph problem, the input is a graph G and an integer $k > 0$. The goal is to compute a subset $S \subseteq V(G)$ of k vertices, maximizing $|E_G(S)|$. We denote an instance of the problem by $\mathrm{DkS}(G,k)$, and we denote the value of the optimal solution to instance $\mathrm{DkS}(G,k)$ by $\mathrm{OPT}_{\mathrm{DkS}}(G,k)$.

We also consider a bipartite version of the Densest k-Subgraph problem, called Bipartite Densest (k_1,k_2)-Subgraph. This problem was first studied in [2]. The input to the problem is a bipartite graph $G = (A,B,E)$ and positive integers k_1,k_2. The goal is to compute a subset $S \subseteq V(G)$ of vertices with $|S \cap A| = k_1$ and $|S \cap B| = k_2$, such that $|E_G(S)|$ is maximized. An instance of this problem is denoted by $\mathrm{BDkS}(G,k_1,k_2)$, and the value of the optimal solution to instance $\mathrm{BDkS}(G,k_1,k_2)$ is denoted by $\mathrm{OPT}_{\mathrm{BDkS}}(G,k_1,k_2)$. The following lemma shows that the Bipartite Densest (k_1,k_2)-Subgraph problem and the Densest k-Subgraph problem are roughly equivalent from the approximation viewpoint. Similar results were also shown in prior work.

▶ **Lemma 2.** *Let $\alpha : \mathbb{Z}^+ \to \mathbb{Z}^+$ be an increasing function such that $\alpha(n) = o(n)$. Then the following hold:*

- *If there exists an $\alpha(n)$-approximation algorithm for the* Densest k-Subgraph *problem with running time at most $T(n)$, where n is the number of vertices in the input graph, then there exists an $O(\alpha(N^2))$-approximation algorithm for the* Bipartite Densest (k_1,k_2)-Subgraph *problem, with running time $O(T(N^2) \cdot \mathrm{poly}(N))$, where N is the number of vertices in the input graph. Moreover, if the algorithm for* Densest k-Subgraph *is deterministic, then so is the algorithm for* Bipartite Densest (k_1,k_2)-Subgraph.
- *Similarly, if there exists an efficient $\alpha(N)$-approximation algorithm for the* Bipartite Densest (k_1,k_2)-Subgraph *problem, where N is the number of vertices in the input graph, then there exists an efficient $O(\alpha(2n))$-approximation algorithm for the* Densest k-Subgraph *problem, where n is the number of vertices in the input graph. Moreover, if the algorithm for* Bipartite Densest (k_1,k_2)-Subgraph *is deterministic, then so is the algorithm for* Densest k-Subgraph.

Dense k-Coloring.

The input to the Dense k-Coloring problem consists of an n-vertex graph G and an integer $k > 0$, such that n is an integral multiple of k. The goal is to compute a partition of $V(G)$ into n/k subsets $S_1,\ldots,S_{n/k}$ of cardinality k each, while maximizing $\sum_{i=1}^{n/k} |E_G(S_i)|$. An instance of the Dense k-Coloring problem is denoted by $\mathrm{DkC}(G,k)$, and the value of the optimal solution to instance $\mathrm{DkC}(G,k)$ is denoted by $\mathrm{OPT}_{\mathrm{DkC}}(G,k)$.

(r, h)-Graph Partitioning.

The input to the (r,h)-Graph Partitioning problem consists of a graph G, and integers $r, h > 0$. The goal is to compute r vertex-disjoint subgraphs $H_1, \ldots, H_r$ of G, such that for all $1 \le i \le r$, $|E(H_i)| \le h$, while maximizing $\sum_{i=1}^{r} |E(H_i)|$. An instance of the (r,h)-Graph Partitioning problem is denoted by $\mathrm{GP}(G, r, h)$, and the value of the optimal solution to instance $\mathrm{GP}(G, r, h)$ is denoted by $\mathrm{OPT}_{\mathrm{GP}}(G, r, h)$.

Maximum Bounded-Crossing Subgraph.

In the Maximum Bounded-Crossing Subgraph problem, the input is a graph G and an integer $L > 0$. The goal is to compute a subgraph $H \subseteq G$ with $\mathsf{CrN}(H) \le L$, while maximizing $|E(H)|$. An instance of the Maximum Bounded-Crossing Subgraph problem is denoted by $\mathrm{MBCS}(G, L)$, and the value of the optimal solution to instance $\mathrm{MBCS}(G, L)$ is denoted by $\mathrm{OPT}_{\mathrm{MBCS}}(G, L)$. We note that we can assume that $L \le |V(G)|^4$, as otherwise the optimal solution is the whole graph G, since the crossing number of a simple graph G is at most $|E(G)|^2 \le |V(G)|^4$.

3 Conditional Hardness of Densest k-Subgraph

3.1 Conjecture on Hardness of 2-CSP's

We consider the Bipartite 2-CSP problem, that is defined as follows. The input to the problem consists of two sets X, Y of variables, together with an integer $A > 1$. Every variable $z \in X \cup Y$ takes values in set $[A] = \{1, \ldots, A\}$. We are also given a collection $\mathcal{C}$ of constraints, where each constraint $C(x, y) \in \mathcal{C}$ is defined over a pair of variables $x \in X$ and $y \in Y$. For each such constraint, we are given a truth table that, for every pair of assignments a to x and a' to y, specifies whether (a, a') *satisfy* constraint $C(x, y)$. The *value* of the CSP is the largest fraction of constraints that can be simultaneously satisfied by an assignment to the variables.

We associate with each constraint $C = C(x, y) \in \mathcal{C}$, a bipartite graph $G_C = (L, R, E)$, where $L = R = [A]$, and there is an edge (a, a') in E iff the assignments a to x and a' to y satisfy C. Notice that instance $\mathcal{I}$ of the Bipartite 2-CSP problem is completely determined by $X, Y, A, \mathcal{C}$, and the graphs in $\{G_C\}_{C \in \mathcal{C}}$, so we will denote $\mathcal{I} = (X, Y, A, \mathcal{C}, \{G_C\}_{C \in \mathcal{C}})$. The *size* of instance $\mathcal{I}$ is defined to be $\mathrm{size}(\mathcal{I}) = |\mathcal{C}| \cdot A^2 + |X| + |Y|$.

Consider some instance $\mathcal{I} = (X, Y, A, \mathcal{C}, \{G_C\}_{C \in \mathcal{C}})$ of Bipartite 2-CSP. We say that $\mathcal{I}$ is a *d-to-d instance* if, for every constraint C, every vertex of graph $G_C = (L, R, E)$ has degree at most d.

Consider now some functions $d(n), s(n) : \mathbb{R}^+ \to \mathbb{R}^+$. We assume that, for all n, $d(n) \ge 1$ and $s(n) < 1$. In a $(d(n), s(n))$-2CSP problem, the input is an instance $\mathcal{I} = (X, Y, A, \mathcal{C}, \{G_C\}_{C \in \mathcal{C}})$ of Bipartite 2-CSP, such that, if we denote by $n = \mathrm{size}(\mathcal{I})$, then the instance is $d(n)$-to-$d(n)$. We say that $\mathcal{I}$ is a YES-INSTANCE, if there is some assignment to the variables of $X \cup Y$ that satisfies at least $|\mathcal{C}|/2$ of the constraints, and we say that it is a NO-INSTANCE, if the largest number of constraints of $\mathcal{C}$ that can be simultaneously satisfied by any assignment is at most $s(n) \cdot |\mathcal{C}|$. Given an instance $\mathcal{I}$ of $(d(n), s(n))$-2CSP problem, the goal is to distinguish between the case where $\mathcal{I}$ is a YES-INSTANCE and the case where $\mathcal{I}$ is a NO-INSTANCE. If $\mathcal{I}$ is neither a YES-INSTANCE nor a NO-INSTANCE, the output of the algorithm can be arbitrary. We now state our conjecture regarding hardness of $(d(n), s(n))$-2CSP, that is a restatement of Conjecture 1 from the Introduction.

▶ **Conjecture 3.** *There is a constant $0 < \varepsilon \leq 1/2$, such that the* $(d(n), s(n))$-2CSP *problem is* NP*-hard for* $d(n) = 2^{(\log n)^{\varepsilon}}$ *and* $s(n) = 1/2^{64(\log n)^{1/2+\varepsilon}}$.

3.2 Conditional Hardness of Densest k-Subgraph

In the remainder of this section, we prove the following theorem on the conditional hardness of Densest k-Subgraph.

▶ **Theorem 4.** *Assume that Conjecture 3 holds and that* P $\neq$ NP*. Then for some* $0 < \varepsilon \leq 1/2$*, there is no efficient approximation algorithm for* Densest k-Subgraph *problem that achieves approximation factor* $2^{(\log N)^{\varepsilon}}$*, where* N *is the number of vertices in the input graph.*

In fact we will prove a slightly more general theorem, that will be useful for us later.

▶ **Theorem 5.** *Suppose there is an algorithm for the* Densest k-Subgraph *problem, that, given an instance* $\mathrm{DkS}(G, k)$ *with* $|V(G)| = N$*, in time at most* $T(N)$*, computes a factor* $2^{(\log N)^{\varepsilon}}$*-approximate solution to the problem, for some constant* $0 < \varepsilon \leq 1/2$*. Then there is an algorithm, that, given an instance* $\mathcal{I}$ *of* $(d(n), s(n))$-2CSP *problem of size* n*, where* $d(n) = 2^{(\log n)^{\varepsilon}}$ *and* $s(n) = 1/2^{64(\log n)^{1/2+\varepsilon}}$*, responds "YES" or "NO", in time* $O(\mathrm{poly}(n) \cdot T(\mathrm{poly}(n)))$*. If* $\mathcal{I}$ *is a* Yes-Instance*, the algorithm is guaranteed to respond "YES", and if it is a* No-Instance*, it is guaranteed to respond "NO".*

Theorem 4 immediately follows from Theorem 5. The remainder of this section is dedicated to proving Theorem 5. A central notion that we use is a *constraint graph* that is associated with an instance $\mathcal{I}$ of 2-CSP.

Let $\mathcal{I} = (X, Y, A, \mathcal{C}, \{G_C\}_{C \in \mathcal{C}})$ be an instance of the Bipartite 2-CSP problem. The *constraint graph* associated with instance $\mathcal{I}$ is denoted by $H(\mathcal{I})$, and it is defined as follows. The set of vertices of $H(\mathcal{I})$ is the union of two subsets: set $V = \{v(x) \mid x \in X\}$ of vertices representing the variables of X, and set $U = \{v(y) \mid y \in Y\}$ of vertices representing the variables of Y. For convenience, we will not distinguish between the vertices of V and the variables of X, so we will identify each variable $x \in X$ with its corresponding vertex $v(x)$. Similarly, we will not distinguish between vertices of U and variables of Y. The set of edges of $H(\mathcal{I})$ contains, for every constraint $C = C(x, y) \in \mathcal{C}$, edge $e_C = (x, y)$. We say that edge e_C *represents* the constraint C. Notice that, if E' is a subset of edges of $H(\mathcal{I})$, then we can define a set $\Phi(E') \subseteq \mathcal{C}$ of constraints that the edges of E' represent, namely: $\Phi(E') = \{C \in \mathcal{C} \mid e_C \in E'\}$. Next, we define bad sets of constraints and bad sets of edges.

▶ **Definition 6** (Bad Set of Constraints and Bad Collection of Edges)**.** *Let* $\mathcal{C}' \subseteq \mathcal{C}$ *be a collection of constraints of an instance* $\mathcal{I} = (X, Y, A, \mathcal{C}, \{G_C\}_{C \in \mathcal{C}})$ *of Bipartite 2-CSP. We say that* $\mathcal{C}'$ *is a* bad set of constraints *if the largest number of constraints of* $\mathcal{C}'$ *that can be simultaneously satisfied by any assignment to the variables of* $X \cup Y$ *is at most* $\frac{|\mathcal{C}'|}{4}$*. If* $E' \subseteq E(H(\mathcal{I}))$ *is a set of edges of* $H(\mathcal{I})$*, whose corresponding set* $\Phi(E')$ *of constraints is bad, then we say that* E' *is a* bad collection of edges.

The next observation easily follows from the definition of a bad set of constraints.

▶ **Observation 7.** *Let* $\mathcal{I} = (X, Y, A, \mathcal{C}, \{G_C\}_{C \in \mathcal{C}})$ *be an instance of bipartite 2-CSP, and let* $\mathcal{C}', \mathcal{C}'' \subseteq \mathcal{C}$ *be two disjoint sets of constraints that are both bad. Then* $\mathcal{C}' \cup \mathcal{C}''$ *is also a bad set of constraints.*

Next, we define good subsets of constraints and good subgraphs of the constraint graph $H(\mathcal{I})$.

▶ **Definition 8** (Good Set of Constraints and Good Subgraphs of $H(\mathcal{I})$). *Let $\mathcal{C}' \subseteq \mathcal{C}$ be a collection of constraints of an instance $\mathcal{I} = (X, Y, A, \mathcal{C}, \{G_C\}_{C\in\mathcal{C}})$ of Bipartite 2-CSP, and let $0 < \beta \leq 1$ be a parameter. We say that $\mathcal{C}'$ is a* β-good set of constraints, *if there is an assignment to variables of $X \cup Y$ that satisfies at least $\frac{|\mathcal{C}'|}{\beta}$ constraints of $\mathcal{C}'$. If $E' \subseteq E(H(\mathcal{I}))$ is a set of edges of $H(\mathcal{I})$, whose corresponding set $\Phi(E')$ of constraints is β-good, then we say that E' is a* β-good collection of edges. *Lastly, if $H' \subseteq H(\mathcal{I})$ is a subgraph of the constraint graph, and the set $E(H')$ of edges is β-good, then we say that graph H' is* β-good.

The next observation easily follows from the definition of a good set of constraints.

▶ **Observation 9.** *Let $\mathcal{I} = (X, Y, A, \mathcal{C}, \{G_C\}_{C\in\mathcal{C}})$ be an instance of bipartite 2-CSP, let $0 < \beta \leq 1$ be a parameter, and let H', H'' be two subgraphs of $H(\mathcal{I})$ that are both β-good and disjoint in their vertices. Then graph $H' \cup H''$ is also β-good.*

The observation follows from the fact that, since graphs H', H'' are disjoint in their vertices, if we let $\mathcal{C}' = \Phi(E(H'))$, $\mathcal{C}'' = \Phi(E(H''))$ be the sets of constraints associated with the edge sets of both graphs, then the variables participating in the constraints of $\mathcal{C}'$ are disjoint from the variables participating in the constraints of $\mathcal{C}''$.

The following theorem is key in proving Theorem 5.

▶ **Theorem 10.** *Assume that there exists a constant $0 < \varepsilon \leq 1/2$, and an $\alpha(N)$-approximation algorithm $\mathcal{A}$ for the* Densest k-Subgraph *problem, whose running time is at most $T(N)$, where N is the number of vertices in the input graph, and $\alpha(N) = 2^{(\log N)^{\varepsilon}}$. Then there is an algorithm, whose input consists of an instance $\mathcal{I} = (X, Y, A, \mathcal{C}, \{G_C\}_{C\in\mathcal{C}})$ of Bipartite 2-CSP and parameter n that is greater than a large enough constant, so that $\mathrm{size}(\mathcal{I}) \leq n$ holds, and $\mathcal{I}$ is a $d(n)$-to-$d(n)$ instance of Bipartite 2-CSP, for $d(n) \leq 2^{(\log n)^{\varepsilon}}$. Let $\beta = 2^{8(\log n)^{1/2+\varepsilon}}$, and let $r = \lceil \beta \cdot \log n \rceil$. The algorithm returns a partition $(E^b, E_1, \ldots, E_r)$ of $E(H(\mathcal{I}))$, such that E^b is a bad set of edges, and for all $1 \leq i \leq r$, set E_i of edges is β^3-good. The running time of the algorithm is $O(T(\mathrm{poly}(n)) \cdot \mathrm{poly}(n)$.*

The proof of Theorem 5 easily follows from Theorem 10. Assume that there exists a constant $0 < \varepsilon \leq 1/2$, and an $\alpha(N)$-approximation algorithm $\mathcal{A}$ for the Densest k-Subgraph problem, whose running time is at most $T(N)$, where N is the number of vertices in the input graph, and $\alpha(N) = 2^{(\log N)^{\varepsilon}}$. We show an algorithm for the $(d(n), s(n))$-2CSP problem, for $d(n) = 2^{(\log n)^{\varepsilon}}$ and $s(n) = 1/2^{64(\log n)^{1/2+\varepsilon}}$. Let $\mathcal{I} = (X, Y, A, \mathcal{C}, \{G_C\}_{C\in\mathcal{C}})$ be an input instance of the Bipartite 2-CSP problem, with $\mathrm{size}(\mathcal{I}) = n$, so that $\mathcal{I}$ is a $d(n)$-to-$d(n)$ instance of Bipartite 2-CSP, for $d(n) \leq 2^{(\log n)^{\varepsilon}}$. If n is bounded by a constant, then we can determine whether $\mathcal{I}$ is a YES-INSTANCE or a NO-INSTANCE by exhaustively trying all assignments to its variables. Therefore, we assume that n is greater than a large enough constant. We apply the algorithm from Theorem 10 to this instance $\mathcal{I}$. Let $(E^b, E_1, \ldots, E_r)$ be the partition of the edges of $E(H(\mathcal{I}))$ that the algorithm returns. We now consider two cases.

Assume first that $|E^b| > 2|\mathcal{C}|/3$. Let $\mathcal{C}^b \subseteq \mathcal{C}$ be the set of all constraints that correspond to the edges of E^b. Recall that set $\mathcal{C}^b$ of constraints is bad, so in any assignment, at most $\frac{|\mathcal{C}^b|}{4}$ of the constraints in $\mathcal{C}^b$ may be satisfied. Therefore, if f is any assignment to variables of $X \cup Y$, the number of constraints in $\mathcal{C}$ that are not satisfied by f is at least $\frac{3|\mathcal{C}^b|}{4} > \frac{|\mathcal{C}|}{2}$. Clearly, $\mathcal{I}$ may not be a YES-INSTANCE in this case. Therefore, if $|E^b| > 2|\mathcal{C}|/3$, we report that $\mathcal{I}$ is a NO-INSTANCE.

If $|E^b| \leq 2|\mathcal{C}|/3$, then we report that $\mathcal{I}$ is a YES-INSTANCE. It is now enough to show that, if $|E^b| \leq 2|\mathcal{C}|/3$, then instance $\mathcal{I}$ may not be a NO-INSTANCE. In other words, it is enough to show that there is an assignment that satisfies more than $\frac{|\mathcal{C}|}{2^{64(\log n)^{1/2+\varepsilon}}}$ constraints.

Indeed, since $|E^b| \leq 2|\mathcal{C}|/3$, there is an index $1 \leq i \leq r$, with $|E_i| \geq \frac{|\mathcal{C}|}{3r}$. Since set E_i of edges is β^3-good, there is an assignment to the variables of $X \cup Y$, that satisfies at least $\frac{|E_i|}{\beta^3} \geq \frac{|\mathcal{C}|}{3r\beta^3}$ constraints that correspond to the edges of E_i. Recall that $\beta = 2^{8(\log n)^{1/2+\varepsilon}}$ and $r = \lceil \beta \cdot \log n \rceil$. Therefore, $3r\beta^3 \leq 6\beta^4 \log n \leq 2^{64(\log n)^{1/2+\varepsilon}}$. We conclude that there is an assignment satisfying at least $|\mathcal{C}|/2^{64(\log n)^{1/2+\varepsilon}}$ constraints, and so $\mathcal{I}$ may not be a No-Instance. It is easy to verify that the running time of the algorithm is $O(T(\text{poly}(n)) \cdot \text{poly}(n)$.

To conclude, we have shown that, if there is an $\alpha(N)$-approximation algorithm $\mathcal{A}$ for the Densest k-Subgraph problem, with running time at most $T(N)$, where N is the number of vertices in the input graph, and $\alpha(N) = 2^{(\log N)^{\varepsilon}}$, then there is an algorithm for the $(d(n), s(n))$-2CSP problem, for $d(n) = 2^{(\log n)^{\varepsilon}}$ and $s(n) = 1/2^{8(\log n)^{1/2+\varepsilon}}$, whose running time is $O(T(\text{poly}(n)) \cdot \text{poly}(n)$.

In the remainder of this section we prove Theorem 10.

3.3 Proof of Theorem 10

The following theorem is the main technical ingredient of the proof of Theorem 10.

▶ **Theorem 11.** *Assume that there exists an $\alpha(N)$-approximation algorithm $\mathcal{A}$ for the* Bipartite Densest (k_1, k_2)-Subgraph *problem, whose running time is at most $T(N)$, where N is the number of vertices in the input graph. Then there is an algorithm, that, given an instance $\mathcal{I} = (X, Y, A, \mathcal{C}, \{G_C\}_{C \in \mathcal{C}})$ of Bipartite 2-CSP and parameters $n, \beta \geq 1$, so that $\text{size}(\mathcal{I}) \leq n$, $\beta \geq 2^{30}(\alpha(n))^3(\log n)^{12}$, and $\mathcal{I}$ is a $d(n)$-to-$d(n)$ instance of Bipartite 2-CSP, for some function $d(n)$, in time $O(T(n) \cdot \text{poly}(n))$, does one of the following:*

- *either correctly establishes that graph $H(\mathcal{I})$ is β^3-good; or*
- *computes a bad set $\mathcal{C}' \subseteq \mathcal{C}$ of constraints, with $|\mathcal{C}'| \geq \frac{|\mathcal{C}|}{8 \log^2 n}$; or*
- *computes a subgraph $H' = (X', Y', E')$ of $H(\mathcal{I})$, for which the following hold:*
  - $|X'| \leq \frac{2d(n) \cdot |X|}{\beta}$;
  - $|Y'| \leq \frac{2d(n) \cdot |Y|}{\beta}$; *and*
  - $|E'| \geq \frac{\text{vol}_H(X' \cup Y')}{2048 d(n) \cdot \alpha(n) \cdot \log^4 n}$.

The proof of Theorem 11 partially relies on ideas and techniques from [18], and is deferred to the full version of the paper. We now complete the proof of Theorem 10 using Theorem 11, starting with the following simple corollary, whose proof is deferred to the full version of the paper.

▶ **Corollary 12.** *Assume that there exists an $\alpha(N)$-approximation algorithm $\mathcal{A}$ for the* Bipartite Densest (k_1, k_2)-Subgraph *problem, whose running time is at most $T(N)$, where N is the number of vertices in the input graph. Then there is an algorithm, whose input consists of an instance $\mathcal{I} = (X, Y, A, \mathcal{C}, \{G_C\}_{C \in \mathcal{C}})$ of Bipartite 2-CSP and parameters $n, \beta \geq 1$, so that $\text{size}(\mathcal{I}) \leq n$, $\beta \geq 2^{30}(\alpha(n))^3(\log n)^{12}$, and $\mathcal{I}$ is a $d(n)$-to-$d(n)$ instance of Bipartite 2-CSP. The algorithm returns a partition (E_1, E_2) of $E(H(\mathcal{I}))$, where E_1 is a bad set of edges, and:*

- *either the algorithm correctly certifies that E_2 is a β^3-good set of edges; or*
- *it computes a subgraph $H' = (X', Y', E')$ of $H(\mathcal{I})$, with $E(H') \subseteq E_2$, for which the following hold:*
  - $|X'| \leq \frac{2d(n) \cdot |X|}{\beta}$;
  - $|Y'| \leq \frac{2d(n) \cdot |Y|}{\beta}$; *and*
  - $|E'| \geq \frac{|E_2^*|}{2048 d(n) \cdot \alpha(n) \cdot \log^4 n}$, *where E_2^* is a set of edges containing every edge $e \in E_2$ with exactly one endpoint in $V(H')$.*

The running time of the algorithm is $O(T(n) \cdot \text{poly}(n))$.

Next, we obtain the following corollary.

▶ **Corollary 13.** *Assume that there exists a constant* $0 < \varepsilon \leq 1/2$*, and an* $\alpha(N)$*-approximation algorithm* $\mathcal{A}$ *for the* Bipartite Densest (k_1, k_2)-Subgraph *problem, whose running time is at most* $T(N)$*, where* N *is the number of vertices in the input graph, and* $\alpha(N) = 2^{(4\log N)^{\varepsilon}}$*. Then there is an algorithm, whose input consists of an instance* $\mathcal{I} = (X, Y, A, \mathcal{C}, \{G_C\}_{C \in \mathcal{C}})$ *of Bipartite 2-CSP and parameter* n *that is greater than a large enough constant, so that* $\text{size}(\mathcal{I}) \leq n$ *holds, and* $\mathcal{I}$ *is a* $d(n)$*-to-*$d(n)$ *instance of Bipartite 2-CSP, for* $d(n) \leq 2^{(\log n)^{\varepsilon}}$*. Let* $\beta = 2^{8(\log n)^{1/2+\varepsilon}}$*. The algorithm returns a partition* (E_1, E_2, E_3) *of* $E(H(\mathcal{I}))$*, where* E_1 *is a bad set of constraints,* E_2 *is a* β^3*-good set of constraints, and* $|E_1 \cup E_2| \geq \frac{|E(H(\mathcal{I}))|}{\beta}$*. The running time of the algorithm is* $O(T(n) \cdot \text{poly}(n))$*.*

Proof: Throughout the proof, we assume that there exists a constant $0 < \varepsilon \leq 1/2$, and an $\alpha(N)$-approximation algorithm $\mathcal{A}$ for the Bipartite Densest (k_1, k_2)-Subgraph problem, whose running time is at most $T(N)$, where N is the number of vertices in the input graph, and $\alpha(N) = 2^{(2\log N)^{\varepsilon}}$. Assume that we are given an instance $\mathcal{I} = (X, Y, A, \mathcal{C}, \{G_C\}_{C \in \mathcal{C}})$ of Bipartite 2-CSP, together with a parameter n that is greater than a large enough constant, so that $\text{size}(\mathcal{I}) \leq n$, and $\mathcal{I}$ is a $d(n)$-to-$d(n)$ instance of Bipartite 2-CSP, for $d(n) \leq 2^{(\log n)^{\varepsilon}}$. For convenience, we denote $H = H(\mathcal{I})$. Our algorithm uses a parameter $\eta = 2^{12} d(n) \cdot \alpha(n) \cdot \log^4 n$.

The algorithm is iterative. Over the course of the algorithm, we maintain a collection $\mathcal{H}$ of subgraphs of H, and another subgraph H^g of H. We will ensure that, throughout the algorithm, all graphs in $\mathcal{H} \cup \{H^g\}$ are mutually disjoint in their vertices. We denote by $E^g = E(H^g)$ and $E^1 = \bigcup_{H' \in \mathcal{H}} E(H')$. Additionally, we maintain another set E^b of edges of H, that is disjoint from $E^g \cup E^1$, and we denote by $E^0 = E(H) \setminus (E^g \cup E^b \cup E^1)$ the set of all remaining edges of H. We ensure that the following invariants hold throughout the algorithm.

I1. set $E^g = E(H^g)$ of edges is β^3-good;
I2. set E^b of edges is bad; and
I3. all graphs in $\mathcal{H} \cup \{H^g\}$ are disjoint in their vertices.

Intuitively, we will start with the set $\mathcal{H}$ containing a single graph H, and $E^g = E^b = E^0 = \emptyset$. As the algorithm progresses, we will iteratively add edges to sets E^g, E^b and E^0, while partitioning the graphs in $\mathcal{H}$ into smaller subgraphs. The algorithm will terminate once $\mathcal{H} = \emptyset$. The key in the analysis of the algorithm is to ensure that $|E^0|$ is relatively small when the algorithm terminates. We do so via a charging scheme: we assign a budget to every edge of $E^1 \cup E^g \cup E^b$, that evolves over the course of the algorithm, and we keep track of this budget over the course of the algorithm.

In order to define vertex budgets, we will assign, to every graph $H \in \mathcal{H}$ a *level*, that is an integer between 0 and $\lceil \log n \rceil$. We will ensure that, throughout the algorithm, the following additional invariants hold:

I4. If $H' \in \mathcal{H}$ is a level-i graph, then the budget of every edge $e \in E(H')$ is at most η^i; and
I5. Throughout the algorithm's execution, the total budget of all edges in $E^g \cup E^b \cup E^1$ is at least $|E(H)|$.

Intuitively, at the end of the algorithm, we will argue that the level of every graph in $\mathcal{H}$ is not too large, and that the budget of every edge in $E^g \cup E^b \cup E^1$ is not too large. Since the total budget of all edges in $E^g \cup E^b \cup E^1$ is at least $|E(H)|$, it will then follow that $|E^g \cup E^b \cup E^1|$ is sufficiently large. We now proceed to describe the algorithm.

Our algorithm will repeatedly use the algorithm from Corollary 12, with the same functions $\alpha(N), d(n)$, and parameter β. In order to be able to use the corollary, we need to estalish that $\beta \geq 2^{30}(\alpha(n))^3(\log n)^{12}$. This is immediate to verify since $\beta = 2^{8(\log n)^{1/2+\varepsilon}}$, $\alpha(n) = 2^{(4\log n)^{\varepsilon}}$, and n is large enough.

At the beginning of the algorithm, we set $E^0 = E^g = E^b = \emptyset$, and we let $\mathcal{H}$ contain a single graph H, which is assigned level 0. Note that $E^1 = E(H)$ must hold. Every edge $e \in E(H)$ is assigned budget $b(e) = 1$. Clearly, the total budget of all edges of $E^1 \cup E^g \cup E^b$ is $B = \sum_{e \in E^1 \cup E^g \cup E^b} b(e) = |E(H)|$. The algorithm performs iterations, as long as $\mathcal{H} \neq \emptyset$. In every iteration, we select an arbitrary graph $H' \in \mathcal{H}$ to process.

We now describe an iteration where some graph $H' \in \mathcal{H}$ is processed. We assume that graph H' is assigned level i. Notice that graph H' naturally defines an instance $\mathcal{I}' = (X', Y', A, \mathcal{C}', \{G_C\}_{C \in \mathcal{C}'})$ of Bipartite 2-CSP, where $X' = V(H') \cap X$, $Y' = V(H') \cap Y$, $\mathcal{C}' = \{C \in \mathcal{C} \mid e_C \in E(H')\}$, and the graphs G_C for constraints $C \in \mathcal{C}'$ remain the same as in instance $\mathcal{I}$. Clearly, $\text{size}(\mathcal{I}') \leq \text{size}(\mathcal{I}) \leq n$, and $H(\mathcal{I}') = H'$. Furthermore, instance $\mathcal{I}'$ remains a $d(n)$-to-$d(n)$ instance. We apply the algorithm from Corollary 12 to instance $\mathcal{I}'$, with parameters n and β remaining unchanged. Consider the partition (E_1, E_2) of $E(H')$ that the algorithm returns. Recall that the set E_1 of edges is bad. We add the edges of E_1 to set E^b. From Invariant I2 and Observation 7, set E^b of edges continues to be bad. If the algorithm from Corollary 12 certified that E_2 is a β^3-good set of edges, then we update graph H^g to be $H^g \cup (H' \setminus E_1)$, and we add the edges of E_2 to set E^g. We then remove graph H' from $\mathcal{H}$, and continue to the next iteration. Note that, from Observation 9 and Invariants I1 and I3, the set E^g of edges continues to be β^3-good. It is easy to verify that all remaining invariants also continue to hold.

From now on we assume that the algorithm from Corollary 12 returned a subgraph $H'' = (X'', Y'', E'')$ of H', with $E'' \subseteq E_2$, such that $|X''| \leq \frac{2d(n)\cdot|X'|}{\beta}$ and $|Y''| \leq \frac{2d(n)\cdot|Y'|}{\beta}$. In particular, $|V(H'')| = |X''| + |Y''| \leq \frac{2d(n)}{\beta}\cdot(|X'| + |Y'|) \leq \frac{2d(n)}{\beta}\cdot|V(H')|$. Additionally, if we denote by E_2^* the subset of edges of E_2 containing all edges with exactly one endpoint in $X'' \cup Y''$, then $|E''| \geq \frac{|E_2^*|}{2048d(n)\cdot\alpha(n)\cdot\log^4 n}$ must hold. We let H^* be the graph obtained from $H' \setminus E_1$, by deleting the vertices of H'' from it, so $V(H^*) \cup V(H'') = V(H')$, and $E(H^*) \cup E(H'') \cup E_2^* = E_2$. We remove graph H' from $\mathcal{H}$, and we add graphs H'' and H^* to $\mathcal{H}$, with graph H'' assigned level $(i+1)$, and graph H^* assigned level i. We also add the edges of E_2^* to E^0, and we update the set E^1 of edges to contain all edges of $\bigcup_{\tilde{H} \in \mathcal{H}} E(\tilde{H})$. Since we did not modify graph H^g in the current iteration, it is immediate to verify that Invariants I1–I3 continue to hold. Next, we update the budgets of edges, in order to ensure that Invariants I4 and I5 continue to hold. Intuitively, the edges of E_2^* are now added to set E^0, so we need to distribute their budget among the edges of $E(H'')$, in order to ensure that the total budget of all edges in $E^g \cup E^b \cup E^1$ does not decrease. This will ensure that Invariant I5 continues to hold. At the same time, since the level of graph H'' is $(i+1)$, while the level of graph H' was i, we can increase the budgets of the edges of $E(H')$ and still maintain Invariant I4.

Formally, recall that Corollary 12 guarantees that $|E_2^*| \leq |E''| \cdot (2048d(n)\cdot\alpha(n)\cdot\log^4 n) = \frac{|E''|\cdot\eta}{2}$. From Invariant I4, the current budget of every edge in $E'' \cup E_2^*$ is bounded by η^i. Therefore, at the beginning of the current iteration: $\sum_{e \in E'' \cup E_2^*} b(e) \leq \eta^i \cdot (|E_2^*| + |E''|) \leq \eta^i \cdot |E''| \cdot \left(1 + \frac{\eta}{2}\right) < \eta^{i+1} \cdot |E''|$.

We set the budget of every edge in E'' to be η^{i+1}, and leave the budgets of all other edges unchanged. It is easy to verify that $\bigcup_{e \in E^g \cup E^b \cup E^1} b(e)$ does not decrease in the current iteration, so Invariant I5 continues to hold. It is also easy to verify that Invariant I4 continues to hold. Therefore, all invariants continue to hold at the end of the iteration. This completes the description of an iteration.

The algorithm terminates when $\mathcal{H} = \emptyset$. Clearly, we obtain a partition (E^g, E^b, E^0) of $E(H)$ into disjoint subsets, where the set E^b of edges is bad, and the set E^g of edges is β^3-good. It remains to show that $|E^g \cup E^b| \geq \frac{|E(H)|}{\beta}$. We use the edge budgets in order to prove this. Let L^* be the largest level of any subgraph of H that belonged to $\mathcal{H}$ at any time during the algorithm. We start with the following key observation, whose proof is deferred to the full version of the paper.

▶ **Observation 14.** $L^* \leq (\log n)^{1/2-\varepsilon}$.

From Invariant I4, throughout the algorithm, for every edge $e \in E^1$, $b(e) \leq \eta^{L^*}$ must hold. Once an edge is added to $E^b \cup E^g$, its budget does not change. Therefore, at the end of the algorithm, the budget of every edge in $E^g \cup E^b$ is at most η^{L^*}. On the other hand, from Invariant I5, at the end of the algorithm, the total budget of all edges in $E^1 \cup E^g \cup E^b$ is at least $|E(H)|$. Therefore, at the end of the algorithm, $|E^g \cup E^b| \geq \frac{|E(H)|}{\eta^{L^*}}$ holds.

We now bound η^{L^*}. Recall that $\eta = 2^{12} d(n) \cdot \alpha(n) \cdot \log^4 n \leq 2^{4(\log n)^{\varepsilon}}$, since $d(n) \leq 2^{(\log n)^{\varepsilon}}$, $\alpha(n) = 2^{(4 \log n)^{\varepsilon}}$, and n is large enough. Since, from Observation 14, $L^* \leq (\log n)^{1/2-\varepsilon}$, we get that $\eta^{L^*} \leq 2^{4(\log n)^{1/2}} < \beta$, since $\beta = 2^{8(\log n)^{1/2+\varepsilon}}$. Therefore, $|E^g \cup E^b| \geq |E(H)|/\beta$ as required.

Lastly, it is easy to verify that the algorithm has at most $\mathrm{poly}(n)$ iterations, and the running time of each iteration is bounded by $O(T(n) \cdot \mathrm{poly}(n))$, so the total running time of the algorithm is at most $O(T(n) \cdot \mathrm{poly}(n))$. □

We are now ready to complete the proof of Theorem 10. Assume that there exists a constant $0 < \varepsilon \leq 1/2$, and an $\alpha(N)$-approximation algorithm $\mathcal{A}$ for the Densest k-Subgraph problem, whose running time is at most $T(N)$, where N is the number of vertices in the input graph, and $\alpha(N) = 2^{(\log N)^{\varepsilon}}$. From Lemma 2, there exists an $\alpha'(N)$-approximation algorithm $\mathcal{A}$ for the Bipartite Densest (k_1, k_2)-Subgraph problem, where N is the number of vertices in the input graph, and $\alpha'(N) \leq O(\alpha(N^2)) \leq O\left(2^{(2 \log N)^{\varepsilon}}\right)$. The running time of the algorithm is at most $O(T(N^2) \cdot \mathrm{poly}(N))$. Denote by $T'(N) = O(T(N^2) \cdot \mathrm{poly}(N))$ this bound on the running time of the algorithm, and let $\alpha''(N) = 2^{(4 \log N)^{\varepsilon}}$. Then there is an $\alpha''(N)$-approximation algorithm for Bipartite Densest (k_1, k_2)-Subgraph with running time at most $O(T'(N))$.

Assume now that we are given an instance $\mathcal{I} = (X, Y, A, \mathcal{C}, \{G_C\}_{C \in \mathcal{C}})$ of Bipartite 2-CSP and parameter n that is greater than a large enough constant, so that $\mathrm{size}(\mathcal{I}) \leq n$ holds, and $\mathcal{I}$ is a $d(n)$-to-$d(n)$ instance of Bipartite 2-CSP, for $d(n) \leq 2^{(\log n)^{\varepsilon}}$. Let $\beta = 2^{8(\log n)^{1/2+\varepsilon}}$, and let $r = \lceil \beta \cdot \log n \rceil$. For convenience, we denote $H = H(\mathcal{I})$. Initially, we set $E^b = \emptyset$. Our algorithm performs r iterations, where for all $1 \leq j \leq r$, in iteration j we construct the set $E_j \subseteq E(H)$ of edges, that is β^3-good, and possibly adds some edges to set E^b. We ensure that, throughout the algorithm, the set E^b of edges is bad.

We now describe the jth iteration. We assume that sets $E_1, \ldots, E_{j-1}$ of edges of H were already defined. We construct graph H_j, that is obtained from graph H, by deleting the edges of $E_1 \cup \cdots \cup E_{j-1} \cup E^b$ from it. Notice that graph H_j naturally defines an instance $\mathcal{I}_j = (X, Y, A, \mathcal{C}_j, \{G_C\}_{C \in \mathcal{C}_j})$ of Bipartite 2-CSP, with $H_j = H(\mathcal{I}_j)$, where $\mathcal{C}_j = \{C \in \mathcal{C} \mid e_C \in E(H_j)\}$. We apply the algorithm from Corollary 13 to graph H_j, with parameters n, β, and $d(n)$ remaining unchanged. Consider a partition (E^1, E^2, E^3) of $E(H_j)$ that the algorithm returns. We add the edges of E^1 to set E^b. Since both sets of edges are bad, from Observation 7, set E^b of edges continues to be bad. We also set $E_j = E^2$, which is guaranteed to be a β^3-good set of edges. Recall that Corollary 13 also guarantees that $|E^1 \cup E^2| \geq |E(H_j)|/\beta$. We then continue to the next iteration.

Since, from the above discussion, for all $1 \leq j < r$, $|E(H_{j+1})| \leq \left(1 - \frac{1}{\beta}\right)|E(H_j)|$, and since $r = \lceil \beta \cdot \log n \rceil$, at the end of the algorithm, we are guaranteed that the final collection $E^b, E_1, \ldots, E_r$ of subsets of edges indeed partitions $E(H)$.

Notice that the running time of a single iteration is bounded by $O(T'(n) \cdot \text{poly}(n)) \leq O(T(\text{poly}(n)) \cdot \text{poly}(n))$. Since the number of iterations is bounded by $\text{poly}(n)$, the total running time of the algorithm is bounded by $O(T(\text{poly}(n)) \cdot \text{poly}(n))$.

4 Reductions from Dense k-Coloring and (r,h)-Graph Partitioning to Densest k-Subgraph

Our reductions from the Dense k-Coloring and (r,h)-Graph Partitioning problems to Densest k-Subgraph are summarized in the following theorem.

▶ **Theorem 15.** *Let $\alpha : \mathbb{Z}^+ \to \mathbb{Z}^+$ be an increasing function, such that $\alpha(n) \leq o(n)$. Assume that there is an efficient $\alpha(n)$-approximation algorithm for the* Densest k-Subgraph *problem, where n is the number of vertices in the input graph. Then both of the following hold:*

- *there is an efficient randomized algorithm that, given an instance of* Dense k-Coloring *whose graph contains N vertices, with high probability computes an $O(\alpha(N^2) \cdot \text{poly} \log N)$-approximate solution to this instance; and*
- *there is an efficient randomized algorithm that, given an instance of* (r,h)-Graph Partitioning *whose graph contains N vertices, with high probability computes an $O(\alpha(N^2) \cdot \text{poly} \log N)$-approximate solution to this instance.*

The proof of the theorem is deferred to the full version of the paper, due to lack of space. We provide a high-level overview of the proof of the first assertion: a reduction from Dense k-Coloring to Densest k-Subgraph. The proof of the second assertion is similar. We start by considering an LP-relaxation of the Dense k-Coloring problem, whose number of variables is at least $\binom{N}{k}$. Due to this high number of variables, we cannot solve it directly. We first show an algorithm, that, given an approximate fractional solution to this LP-relaxation, whose support size is polynomial in N, computes an approximate integral solution to the Dense k-Coloring problem instance. We then show an efficient algorithm that computes an approximate solution to the LP-relaxation, whose support is relatively small. In order to do so, we design an approximate separation oracle to the dual LP of the LP-relaxation, that relies on an approximation algorithm for Densest k-Subgraph.

5 Reductions from Densest k-Subgraph to Dense k-Coloring and (r,h)-Graph Partitioning

Our reductions from Densest k-Subgraph to Dense k-Coloring and (r,h)-Graph Partitioning are summarized in the following theorem.

▶ **Theorem 16.** *Let $\alpha : \mathbb{Z}^+ \to \mathbb{Z}^+$ be an increasing function with $\alpha(n) \leq o(n)$. Then the following hold:*

- *If there exists an efficient $\alpha(n)$-approximation algorithm $\mathcal{A}$ for the* Dense k-Coloring *problem, where n is the number of vertices in the input graph, then there exists a randomized algorithm for the* Densest k-Subgraph *problem, whose running time is $N^{O(\log N)}$, that with* *high probability computes an $O(\alpha(N^{O(\log N)}) \cdot \log N)$-approximate solution to the input instance of the problem; here N is the number of vertices in the input instance of* Densest k-Subgraph*.*

- *If there exists an efficient $\alpha(n)$-approximation algorithm for the* (r,h)-Graph Partitioning *problem, where n is the number of vertices in the input graph, then there exists a randomized algorithm for the* Densest k-Subgraph *problem, whose running time is $N^{O(\log N)}$, that with high probability computes an $O((\alpha(N^{O(\log N)}))^3 \cdot \log^2 N)$-approximate solution to the input instance of the problem; here N is the number of vertices in the input instance of* Densest k-Subgraph*.*

Due to lack of space, we defer the proof of the theorem to the full version of the paper. The key to both reductions is a randomized algorithm, that, given an instance $\mathrm{DkS}(G,k)$ of the Densest k-Subgraph problem, constructs an auxiliary graph H. Intuitively, if instance $\mathrm{DkS}(G,k)$ of Densest k-Subgraph has a solution of value h, then with high probability, graph H has close to $|V(H)|/k$ subgraphs that contain close to h edges each. On the other hand, there is an algorithm that, given a subgraph $H' \subseteq H$ that contains at most k vertices, extracts a subgraph of the original graph G, containing at most k vertices, and close to $|E(H')|$ edges. If $|V(G)| = N$, then our construction of graph H ensures that $|V(H)| \leq N^{O(\log N)}$, which leads to the quasi-polynomial time of our reductions. The specific construction of the graph H is inspired by the ideas from [34]. We obtain the following immediate corollary of Theorem 16, whose proof is deferred to the full version of the paper due to lack of space.

▶ **Corollary 17.** *Assume that Conjecture 3 holds and that* NP $\not\subseteq$ BPTIME$(n^{O(\log n)})$*. Then for some constant $0 < \varepsilon' \leq 1/2$, there is no efficient $2^{(\log n)^{\varepsilon'}}$-approximation algorithm for* (r,h)-Graph Partitioning*, and there is no efficient $2^{(\log n)^{\varepsilon'}}$-approximation algorithm for* Dense k-Coloring*.*

6 Reductions between (r,h)-Graph Partitioning and Maximum Bounded-Crossing Subgraph

We establish a connection between the (r,h)-Graph Partitioning and Maximum Bounded-Crossing Subgraph problems via the following two theorems.

▶ **Theorem 18.** *Let $\alpha : \mathbb{Z}^+ \to \mathbb{Z}^+$ be an increasing function with $\alpha(n) = o(n)$. Assume that there exists an efficient $\alpha(n)$-approximation algorithm for the* (r,h)-Graph Partitioning *problem, where n is the number of vertices in the input graph. Then there exists an efficient $O(\alpha(N) \cdot \mathrm{poly}\log N)$-approximation algorithm for* Maximum Bounded-Crossing Subgraph*, where N is the number of vertices in the input instance of* Maximum Bounded-Crossing Subgraph*.*

▶ **Theorem 19.** *Let $\alpha : \mathbb{Z}^+ \to \mathbb{Z}^+$ be an increasing function with $\alpha(n) = o(n)$. Assume that there exists an efficient $\alpha(N)$-approximation algorithm for the* Maximum Bounded-Crossing Subgraph *problem, where N is the number of vertices in the input graph. Then there exists an efficient $O((\alpha(n))^2 \cdot \mathrm{poly}\log n)$-approximation algorithm for* (r,h)-Graph Partitioning*, where n is the number of vertices in the input instance of* (r,h)-Graph Partitioning*.*

The proofs of the above two theorems are deferred to the full version of the paper. Both proofs exploit well-known connections between crossing number and graph partitioning, that can be viewed as an extension of the classical Planar Separator Theorem of [45]: namely, if a graph G has a drawing with at most L crossings, then there is a balanced cut in G, containing at most $O\left(\sqrt{L + \Delta \cdot |E(G)|}\right)$ edges, where Δ is maximum vertex degree in G. Another useful fact exploited in the proofs of both these theorems is that any graph G with m edges has a plane drawing with at most m^2 crossings. For example, if $\mathcal{H} = \{H_1, \ldots, H_r\}$ is a solution to an instance of the (r,h)-Graph Partitioning problem on graph G, then there is a drawing of graph $H = \bigcup_{i=1}^{r} H_i$, in which the number of crossings is bounded by $r \cdot h^2$. These two facts are exploited in order to establish a close relationship between the (r,h)-Graph Partitioning and Maximum Bounded-Crossing Subgraph problems, and complete the proofs of Theorems 18 and 19. By combining Theorem 19 with Corollary 17, we obtain the following corollary, whose proof is deferred to the full version of the paper.

▶ **Corollary 20.** *Assume that Conjecture 3 holds and that* $\mathsf{NP} \not\subseteq \mathsf{BPTIME}(n^{O(\log n)})$*. Then for some constant* $0 < \varepsilon' \leq 1/2$*, there is no efficient* $2^{(\log n)^{\varepsilon'}}$*-approximation algorithm for* Maximum Bounded-Crossing Subgraph*.*

7 Acknowledgement

The authors thank Irit Dinur and Uri Feige for insightful and helpful discussions.

References

1. M. Ajtai, V. Chvátal, M. Newborn, and E. Szemerédi. Crossing-free subgraphs. *Theory and Practice of Combinatorics*, pages 9–12, 1982.
2. Noga Alon, Sanjeev Arora, Rajsekar Manokaran, Dana Moshkovitz, and Omri Weinstein. Inapproximabilty of densest k-subgraph from average case hardness. *Manuscript*, 2011. `https://www.tau.ac.il/~nogaa/PDFS/dks8.pdf`.
3. Christoph Ambuhl, Monaldo Mastrolilli, and Ola Svensson. Inapproximability results for sparsest cut, optimal linear arrangement, and precedence constrained scheduling. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 329–337. IEEE, 2007.
4. Sanjeev Arora, Boaz Barak, and David Steurer. Subexponential algorithms for unique games and related problems. *Journal of the ACM (JACM)*, 62(5):1–25, 2015.
5. Boaz Barak, Parikshit Gopalan, Johan Håstad, Raghu Meka, Prasad Raghavendra, and David Steurer. Making the long code shorter. *SIAM Journal on Computing*, 44(5):1287–1324, 2015.
6. Boaz Barak, Pravesh K. Kothari, and David Steurer. Small-set expansion in shortcode graph and the 2-to-2 conjecture. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, pages 9:1–9:12, 2019. URL: `https://doi.org/10.4230/LIPIcs.ITCS.2019.9`, `doi:10.4230/LIPIcs.ITCS.2019.9`.
7. Siddharth Barman. Approximating nash equilibria and dense bipartite subgraphs via an approximate version of caratheodory's theorem. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 361–369, 2015.
8. 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, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 201–210, 2010. URL: `http://doi.acm.org/10.1145/1806689.1806718`, `doi:10.1145/1806689.1806718`.
9. Aditya Bhaskara, Moses Charikar, Venkatesan Guruswami, Aravindan Vijayaraghavan, and Yuan Zhou. Polynomial integrality gaps for strong sdp relaxations of densest k-subgraph. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 388–405. SIAM, 2012.
10. Mark Braverman, Young Kun Ko, Aviad Rubinstein, and Omri Weinstein. Eth hardness for densest-k-subgraph with perfect completeness. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1326–1341. SIAM, 2017.

11 Sergio Cabello. Hardness of approximation for crossing number. *Discrete & Computational Geometry*, 49(2):348–358, 2013.

12 Shih-Chia Chang, Li-Hsuan Chen, Ling-Ju Hung, Shih-Shun Kao, and Ralf Klasing. The hardness and approximation of the densest k-subgraph problem in parameterized metric graphs. In *2020 International Computer Symposium (ICS)*, pages 126–130. IEEE, 2020.

13 Chandra Chekuri and Anastasios Sidiropoulos. Approximation algorithms for euler genus and related problems. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 167–176. IEEE, 2013.

14 Markus Chimani and Petr Hliněnỳ. A tighter insertion-based approximation of the crossing number. In *International Colloquium on Automata, Languages, and Programming*, pages 122–134. Springer, 2011.

15 Eden Chlamtác, Michael Dinitz, Christian Konrad, Guy Kortsarz, and George Rabanca. The densest k-subhypergraph problem. *SIAM Journal on Discrete Mathematics*, 32(2):1458–1477, 2018.

16 Julia Chuzhoy. An algorithm for the graph crossing number problem. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 303–312. ACM, 2011.

17 Julia Chuzhoy. Excluded grid theorem: Improved and simplified. In *Proceedings of the forty-seventh annual ACM symposium on Theory of Computing*, pages 645–654, 2015.

18 Julia Chuzhoy, David Hong Kyun Kim, and Rachit Nimavat. Almost polynomial hardness of node-disjoint paths in grids. *Theory of Computing*, 17(6):1–57, 2021.

19 Julia Chuzhoy, Sepideh Mahabadi, and Zihan Tan. Towards better approximation of graph crossing number. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 73–84. IEEE, 2020. Full version: Arxiv:2011.06545.

20 Julia Chuzhoy, Yury Makarychev, and Anastasios Sidiropoulos. On graph crossing number and edge planarization. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete algorithms*, pages 1050–1069. SIAM, 2011.

21 Julia Chuzhoy and Zihan Tan. A subpolynomial approximation algorithm for graph crossing number in low-degree graphs. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, pages 303–316, 2022. Full version: Arxiv:2202.06827.

22 Irit Dinur, Subhash Khot, Guy Kindler, Dor Minzer, and Muli Safra. On non-optimally expanding sets in Grassmann graphs. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 940–951, 2018. URL: `https://doi.org/10.1145/3188745.3188806`, `doi:10.1145/3188745.3188806`.

23 Irit Dinur, Subhash Khot, Guy Kindler, Dor Minzer, and Muli Safra. Towards a proof of the 2-to-1 games conjecture? In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 376–389, 2018. URL: `https://doi.org/10.1145/3188745.3188804`, `doi:10.1145/3188745.3188804`.

24 Guy Even, Sudipto Guha, and Baruch Schieber. Improved approximations of crossings in graph drawings and vlsi layout areas. *SIAM Journal on Computing*, 32(1):231–252, 2002.

25 Uriel Feige. Relations between average case complexity and approximation complexity. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 534–543, 2002.

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

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

28 Uriel Feige, Michael Seltser, et al. On the densest k-subgraph problem. Technical Report CS97-16, Weizmann Institute of Science., 1997. `https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.37.9962&rep=rep1&type=pdf`.

29 Doron Goldstein and Michael Langberg. The dense k subgraph problem. *arXiv preprint arXiv:0912.5327*, 2009.

30 Tesshu Hanaka. Computing densest k-subgraph with structural parameters. *arXiv preprint arXiv:2207.09803*, 2022.

31 Johan Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4):798–859, 2001.

32 Ken-ichi Kawarabayashi and Anastasios Sidiropoulos. Polylogarithmic approximation for minimum planarization (almost). In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 779–788, 2017. URL: `https://doi.org/10.1109/FOCS.2017.77`, `doi:10.1109/FOCS.2017.77`.

33 Ken-ichi Kawarabayashi and Anastasios Sidiropoulos. Polylogarithmic approximation for euler genus on bounded degree graphs. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 164–175. ACM, 2019.

34 Sanjeev Khanna, Nathan Linial, and Shmuel Safra. On the hardness of approximating the chromatic number. *Combinatorica*, 20(3):393–415, 2000.

35 Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 767–775, 2002.

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

37 Subhash Khot, Dor Minzer, Dana Moshkovitz, and Muli Safra. Small set expansion in the Johnson graph. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:78, 2018. `https://eccc.weizmann.ac.il/report/2018/078`.

38 Subhash Khot, Dor Minzer, and Muli Safra. On independent sets, 2-to-2 games, and Grassmann graphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 576–589, 2017. URL: `https://doi.org/10.1145/3055399.3055432`, `doi:10.1145/3055399.3055432`.

39 Subhash Khot, Dor Minzer, and Muli Safra. Pseudorandom sets in Grassmann graph have near-perfect expansion. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 592–601, 2018. URL: `https://doi.org/10.1109/FOCS.2018.00062`, `doi:10.1109/FOCS.2018.00062`.

40 Subhash Khot and Muli Safra. A two-prover one-round game with strong soundness. *Theory of Computing*, 9:863–887, 2013. URL: `https://doi.org/10.4086/toc.2013.v009a028`, `doi:10.4086/toc.2013.v009a028`.

41 G Kortsarz and D Peleg. On choosing a dense subgraph. In *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science*, pages 692–701. IEEE Computer Society, 1993.

42 F. T. Leighton. *Complexity issues in VLSI: optimal layouts for the shuffle-exchange graph and other networks*. MIT Press, 1983.

43 Tom Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM (JACM)*, 46(6):787–832, 1999.

44 Bingkai Lin. The parameterized complexity of the k-biclique problem. *Journal of the ACM (JACM)*, 65(5):1–23, 2018.

45 Richard J Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.

46 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, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 954–961, 2017. URL: `http://doi.acm.org/10.1145/3055399.3055412`, `doi:10.1145/3055399.3055412`.

47 Pasin Manurangsi. Inapproximability of maximum biclique problems, minimum k-cut and densest at-least-k-subgraph from the small set expansion hypothesis. *Algorithms*, 11(1):10, 2018.

48 J. Matoušek. *Lectures on discrete geometry*. Springer-Verlag, 2002.

49 J. Pach and G. Tóth. Thirteen problems on crossing numbers. *Geombinatorics*, 9(4):194–207, 2000.

50 R. B. Richter and G. Salazar. Crossing numbers. In L. W. Beineke and R. J. Wilson, editors, *Topics in Topological Graph Theory*, chapter 7, pages 133–150. Cambridge University Press, 2009.

51 Marcus Schaefer. The graph crossing number and its variants: A survey. *The electronic journal of combinatorics*, pages DS21–Sep, 2012.

52 Renata Sotirov. On solving the densest k-subgraph problem on large graphs. *Optimization Methods and Software*, 35(6):1160–1178, 2020.

53 David Steurer. Subexponential algorithms for d-to-1 two-prover games and for certifying almost perfect expansion. Available at `https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.189.5388&rep=rep1&type=pdf`, 2010.

54 P. Turán. A note of welcome. *J. Graph Theory*, 1:1–5, 1977.