A New Conjecture on Hardness of Low-Degree 2-CSPs with Implications to Hardness of Densest k-Subgraph and Other Problems

Julia Chuzhoy* Mina Dalirrooyfard† Vadim Grinberg‡ Zihan Tan§

November 14, 2022

Abstract

We propose a new conjecture on hardness of low-degree 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 further 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.

*Toyota Technological Institute at Chicago. Email: cjulia@ttic.edu. Supported in part by NSF grant CCF-2006464.

†Massachusetts Institute of Technology. Email: minad@mit.edu. Part of the work was done while the author was a summer intern at TTIC.

‡Weizmann Institute of Science. Email: vadim.grinberg@weizmann.ac.il.

§DIMACS, Rutgers University. Email: zihantan1993@gmail.com. Supported by a grant to DIMACS from the Simons Foundation (820931). Work done while the author was a graduate student at University of Chicago.

Contents

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, that we call Low-Degree CSP Conjecture, 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 Low-Degree CSP 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 [CKN21], 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 [CKN21]. 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 the Low-Degree CSP Conjecture 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. [KP93, FS$^+$97, FPK01, FL01, Fei02, Kho06, GL09, BCC$^+$10, AAM$^+$11, BCG$^+$12, Bar15, BKRW17, Man17, CDK$^+$18, Man18, Lin18, Sot20, CCH$^+$20, Han22]). At the same time it seems notoriously difficult, and despite this extensive work, our understanding of its approximability is still incomplete. The best current approximation algorithm for Densest k-Subgraph, due to [BCC$^+$10], 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 P $\neq$ NP assumption, or even the stronger assumption that NP $\not\subseteq$ BPTIME($n^{\text{poly}\log n}$). In a breakthrough result, Khot [Kho06] proved a factor-c hardness of approximation for Densest k-Subgraph, for some small constant c, assuming that NP $\not\subseteq \cap_{\varepsilon>0}$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 [Fei02, AAM+11]. Additionally, a factor $2^{\Omega(\log^{2/3} n)}$-hardness of approximation was shown under assumptions on solving Planted Clique [AAM+11]. In a recent breakthrough, Manurangsi [Man17] 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 [Man17] can yield such a result. In this paper we show that, assuming the Low-Degree CSP Conjecture 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 [CKN21] 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 $1 \leq i \leq r$, $|E(H_i)| \leq h$, 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 [CKN21] 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, [CKN21] 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 the hardness of (r,h)-Graph Partitioning from existing hardness results for CSP's.

It was noted in [CKN21] 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 [CKN21] 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 [CKN21] for proving hardness of (r,h)-Graph Partitioning with Bundles. We also provide "almost" approximation-preserving reductions between (r,h)-Graph Partitioning to 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 $O(\alpha(n^2)\cdot \operatorname{poly}\log n)$-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 Low-Degree CSP Conjecture, for some constant $0<\varepsilon\leq 1/2$, there is no efficient $2^{(\log n)^{\varepsilon}}$-approximation algorithm for (r,h)-Graph Partitioning, unless NP $\subseteq$ 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 [Tur77] in 1944, and has been extensively studied since then (see, e.g., [Chu11, CMS11, CH11, CS13, KS17, KS19, CMT20], and also [RS09, PT00, Mat02, Sch12] for excellent surveys). But despite all this work, most aspects of the problem are still poorly understood. A long line of work [LR99, EGS02, CMS11, Chu11, KS17, KS19, Chu15, CT22] has recently led to the first sub-polynomial approximation algorithm for the problem in *low degree graphs*. Specifically, [CT22] 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 [Cab13, AMS07]. 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 $|E(H)|$. 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 [ACNS82,Lei83], if $|E(G)| \geq 4|V(G)|$, 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 [CKN21] 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 $O(\alpha(n) \cdot \operatorname{poly}\log n)$-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 $O((\alpha(n))^2 \cdot \operatorname{poly}\log n)$-approximation algorithm for (r,h)-Graph Partitioning. Combined with our conditional hardness of approximation for (r,h)-Graph Partitioning, we get that, assuming the Low-Degree CSP Conjecture, for some constant $0 < \varepsilon \leq 1/2$ there is no efficient $2^{(\log n)^{\varepsilon}}$-approximation algorithm for Maximum Bounded-Crossing Subgraph, unless NP $\subseteq$ 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 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 Low-Degree CSP Conjecture, for some constant $0 < \varepsilon \leq 1/2$, there is no efficient $2^{(\log n)^{\varepsilon}}$-approximation algorithm for Dense k-Coloring, unless

$\mathsf{NP} \subseteq \mathsf{BPTIME}(n^{O(\log n)})$.

The Low-Degree CSP Conjecture. We now turn to describe our new conjecture on the 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, specifies 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 [Kho02] 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 [ABS15] 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 [Kho02]. 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 [Hås01] 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 [ABS15, Ste] 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 [BGH$^+$15, KS13, KMMS18, BKS19, KMS17, DKK$^+$18a, DKK$^+$18b, KMS18].

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, that we refer to as Low-Degree CSP Conjecture, regarding the hardness of Gap-CSP in d-to-d instances.

Conjecture 1 (Low-Degree CSP Conjecture)**.** *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 [Hås01] 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 $n = N^{O(\ell)}$. 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 the Low-Degree CSP Conjecture 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 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 the Low-Degree CSP Conjecture 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 Conjecture 1 holds and 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 [CKN21]. 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 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. 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 $|E(H)|$. This algorithm for decomposing graph H into subgraphs and its analysis employ some of the techniques and ideas introduced in [CKN21], 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 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 an efficient $O(\alpha(n^2) \cdot \operatorname{poly}\log n)$-approximation algorithm for Dense k-Coloring, and an 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 $\text{poly}(n)$, 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 $\text{poly}(n)$ 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 the Low-Degree CSP Conjecture, 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 [KLS00], 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 [KLS00] 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 $O(\alpha(n) \cdot \operatorname{poly}\log n)$-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 $O((\alpha(n))^2 \cdot \operatorname{poly}\log n)$-approximation algorithm for (r,h)-Graph Partitioning. Combined with our conditional hardness of approximation for (r,h)-Graph Partitioning, we get that, assuming the Low-Degree CSP Conjecture, for some constant $0 < \varepsilon \leq 1/2$, there is no efficient $2^{(\log n)^{\varepsilon}}$-approximation algorithm for (r,h)-Graph Partitioning, unless NP $\subseteq$ 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 [LT79]. 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 $\mathcal{H} = \{H_1, \ldots, H_r\}$ 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 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 that show 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 one considers sufficiently large approximation factors, and allows randomized quasi-polynomial time algorithms). We also obtain conditional hardness of approximation results for all four problems based on the Low-Degree CSP Conjecture.

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.

2 Preliminaries

By default, all logarithms are to the base of 2. For a positive integer N, we denote by $[N] = \{1, 2, \ldots, N\}$. 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 [AAM+11]. 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. For completeness, we provide the proof in Appendix A.

Lemma 2.1. *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 $1 \leq i \leq r$, $|E(H_i)| \leq h$, 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 $\mathsf{CrN}(H) \leq L$, 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 $L \leq |V(G)|^4$, as otherwise the optimal solution is the whole graph G, since the crossing number of a simple graph G is at most $|E(G)|^2 \leq |V(G)|^4$.

2.3 Chernoff Bound

We use the following standard version of Chernoff Bound (see. e.g., [DP09]).

Lemma 2.2 (Chernoff Bound)**.** *Let $X_1, \ldots, X_n$ be independent randon variables taking values in $\{0, 1\}$. Let $X = \sum_{1 \leq i \leq n} X_i$, and let $\mu = \mathbf{E}[X]$. Then for any $t > 2e\mu$,*

$$\Pr\left[X > t\right] \leq 2^{-t}.$$

Additionally, for any $0 \leq \delta \leq 1$,

$$\Pr\left[X < (1-\delta) \cdot \mu\right] \leq e^{-\frac{\delta^2 \cdot \mu}{2}}.$$

2.4 Auxiliary Lemma

We use the following simple lemma.

Lemma 2.3. *There is an efficient algorithm, that, given a graph G, a subset S of its vertices, and a parameter $\frac{2}{|S|} < \beta < 1$, computes a set $S' \subseteq S$ of vertices, such that $|S'| \leq \beta \cdot |S|$, and $|E_G(S')| \geq \Omega(\beta^2 \cdot |E_G(S)|)$ holds.*

Proof: Consider the graph $G' = G[S]$ and denote $|S| = k$. We iteratively remove the lowest-degree vertex from G', until G' contains $\lfloor \beta k \rfloor$ vertices. Once the algorithm terminates, we output $S' = V(G')$. It now remains to show that $|E_{G'}(S')| \geq \Omega(\beta^2 |E(G')|)$.

Observe that, if H is an n-vertex graph, and v is a lowest-degree vertex of H, then the degree of v in H is at most $2|E(H)|/n$. Therefore, if vertex v is removed from H, then $|E(H)|$ decreases by at most a factor $(1 - 2/n)$. Therefore,

$$\frac{|E_{G'}(S')|}{|E(G')|} \geq \left(1 - \frac{2}{k}\right)\left(1 - \frac{2}{k-1}\right) \cdots \left(1 - \frac{2}{\lfloor \beta k \rfloor + 1}\right) = \frac{\lfloor \beta k \rfloor \cdot (\lfloor \beta k \rfloor - 1)}{k \cdot (k-1)} = \Omega(\beta^2).$$

□

3 Conditional Hardness of Densest k-Subgraph

3.1 Low-Degree CSP Conjecture

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 defined by $X, Y, A, \mathcal{C}$, and the graphs in $\{G_C\}_{C \in \mathcal{C}}$, so we will denote $\mathcal{I} = (X, Y, A, \mathcal{C}, \{G_C\}_{C \in \mathcal{C}})$. The *size* of instance $\mathcal{I}$ is defined to be $\text{size}(\mathcal{I}) = |\mathcal{C}| \cdot A^2 + |X| + |Y|$.

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, $d(n) \geq 1$ and $s(n) < 1$. In a $(d(n), s(n))$-LD-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 $n = \text{size}(\mathcal{I})$, 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 $(d(n), s(n))$-LD-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 $(d(n), s(n))$-LD-2CSP, that is a restatement of Conjecture 1 from the Introduction.

Conjecture 2 (Low-Degree CSP Conjecture)**.** *There is a constant $0 < \varepsilon \leq 1/2$, such that the $(d(n), s(n))$-LD-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 3.1. *Assume that Conjecture 2 holds and that $\mathsf{P} \neq \mathsf{NP}$. 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 3.2. *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 $(d(n), s(n))$-LD-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 3.1 immediately follows from Theorem 3.2. The remainder of this section is dedicated to proving Theorem 3.2. A central notion that we use is a *constraint graph* that is associated with an instance $\mathcal{I}$ of 2-CSP.

Constraint Graph. 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 collections of edges.

Definition 3.3 (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 3.4. *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 3.5 (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 assigmnet 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 3.6. *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 3.2.

Theorem 3.7. *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 $\text{size}(\mathcal{I}) \leq n$ 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 $\beta = 2^{8(\log n)^{1/2+\varepsilon}}$, 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 $O(T(\text{poly}(n)) \cdot \text{poly}(n)$.*

The proof of Theorem 3.2 easily follows from Theorem 3.7. 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 $(d(n), s(n))$-LD-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 $\text{size}(\mathcal{I}) = n$, 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 3.7 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 $\frac{|\mathcal{C}'|}{4}$ of the constraints in $\mathcal{C}$ 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 $\frac{3|\mathcal{C}'|}{4} > \frac{|\mathcal{C}|}{2}$. 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 $O(T(\mathrm{poly}(n)) \cdot \mathrm{poly}(n)$.

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 $(d(n), s(n))$-LD-2CSP problem, for $d(n) = 2^{(\log n)^{\varepsilon}}$ and $s(n) = 1/2^{8(\log n)^{1/2+\varepsilon}}$, whose running time is $O(T(\mathrm{poly}(n)) \cdot \mathrm{poly}(n)$.

In the remainder of this section we prove Theorem 3.7.

3.3 Proof of Theorem 3.7

The following theorem is the main technical ingredient of the proof of Theorem 3.7.

Theorem 3.8. *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 $\mathrm{size}(\mathcal{I}) \leq n$, $\beta \geq 2^{30}(\alpha(n))^3(\log n)^{12}$, and $\mathcal{I}$ is a $d(n)$-to-$d(n)$ instance of Bipartite 2-CSP, for some function $d(n)$, in time $O(T(n) \cdot \mathrm{poly}(n))$, 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*
- $|E'| \geq \frac{\mathrm{vol}_H(X'\cup Y')}{2048d(n)\cdot\alpha(n)\cdot\log^4 n}$.

We prove Theorem 3.8 in Section Section 3.4, after we complete the proof of Theorem 3.7 using it. We start with the following corollary of Theorem 3.8.

Corollary 3.9. *Assume that there exists an $\alpha(N)$-approximation algorithm $\mathcal{A}$ for the BDkS 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 $\mathrm{size}(\mathcal{I}) \leq n$, $\beta \geq 2^{30}(\alpha(n))^3(\log n)^{12}$, 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*
  - $|E'| \geq \frac{|E_2^*|}{2048d(n)\cdot\alpha(n)\cdot\log^4 n}$, *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 $O(T(n)\cdot\mathrm{poly}(n))$.

Proof: The algorithm is iterative. We start with $E_1 = \emptyset$, $E_2 = E(H(\mathcal{I}))$ and $H = H(\mathcal{I})$. We then iterate. In every iteration, we compute a graph $H' = H \setminus E_1$. We denote by $\mathcal{C}' = \Phi(E(H'))$ the set of all constraints of $\mathcal{C}$ corresponding to the edges of H'. Notice that graph H' naturally defines a $d(n)$-to-$d(n)$ instance $\mathcal{I}'$ of Bipartite 2-CSP, whose size is at most n, that corresponds to the subset $\mathcal{C}' \subseteq \mathcal{C}$ of constraints. We apply the algorithm from Theorem 3.8 to instance $\mathcal{I}'$. If the outcome of the algorithm is a bad set $\mathcal{C}'' \subseteq \mathcal{C}'$ of constraints, then we let $\tilde{E} = \{e_C \mid C \in \mathcal{C}''\}$ be the set of edges of H' corresponding to the constraints of $\mathcal{C}''$. We add the edges of $\tilde{E}$ to E_1, remove them from E_2, and continue to the next iteration.

If the algorithm from Theorem 3.8 certifies that graph H' is β^3-good, then we terminate the algorithm with the current partition (E_1, E_2) of $E(H)$, and certify that the set E_2 of edges is β^3-good.

Otherwise, the outcome of the algorithm from Theorem 3.8 must be a subgraph $H'' = (X', Y', E')$ of H', with $|X'| \leq \frac{2d(n)\cdot|X|}{\beta}$ and $|Y'| \leq \frac{2d(n)\cdot|Y|}{\beta}$. The algorithm also guarantees that $|E'| \geq \frac{\mathrm{vol}_{H'}(X'\cup Y')}{2048d(n)\cdot\alpha(n)\cdot\log^4 n}$.

Let E_2^* be the set of edges containing every edge $e \in E_2$ with exactly one endpoint in $V(H'')$. Since $E(H') = E_2$, it is immediate to verify that $|E_2^*| \leq \mathrm{vol}_{H'}(X' \cup Y')$. Therefore, we are guaranteed

that $|E'| \geq \frac{|E_2^*|}{2048d(n)\cdot\alpha(n)\cdot\log^4 n}$. We return the current partition (E_1, E_2) of $E(H)$ and subgraph H'' of $H(\mathcal{I})$, and terminate the algorithm.

It is easy to verify that the algorithm consists of at most $O(\text{poly}(n))$ iterations, and the running time of each iteration is at most $O(T(n)\cdot\text{poly}(n))$. Therefore, the total running time of the algorithm is at most $O(T(n)\cdot\text{poly}(n))$. □

Next, we obtain the following corollary.

Corollary 3.10. *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 3.9, 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.

Initialization. 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 a single iteration.

Iteration description. 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, $\mathrm{size}(\mathcal{I}') \leq \mathrm{size}(\mathcal{I}) \leq n$, and $H(\mathcal{I}') = H'$. Furthermore, instance $\mathcal{I}'$ remains a $d(n)$-to-$d(n)$ instance. We apply the algorithm from Corollary 3.9 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 3.4, set E^b of edges continues to be bad. If the algorithm from Corollary 3.9 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 3.6 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 3.9 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, $|V(H'')| = |X''| + |Y''| \leq \frac{2d(n)}{\beta} \cdot (|X'| + |Y'|) \leq \frac{2d(n)}{\beta} \cdot |V(H')|$. Additinally, 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 $|E''| \geq \frac{|E_2^*|}{2048 d(n) \cdot \alpha(n) \cdot \log^4 n}$ 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 3.9 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:

$$\sum_{e\in E''\cup E_2^*} b(e) \leq \eta^i \cdot \left(|E_2^*| + |E''|\right) \leq \eta^i \cdot |E''| \cdot \left(1+\frac{\eta}{2}\right) < \eta^{i+1} \cdot |E''|.$$

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.

Observation 3.11. $L^* \leq (\log n)^{1/2-\varepsilon}$.

Proof: Consider any graph H'' that was added to set $\mathcal{H}$ at any time during the algorithm's execution, and assume that H'' was assigned level i. Consider the iteration during which H'' was added to $\mathcal{H}$, and let $H' \in \mathcal{H}$ be the graph that was processed during that iteration. We refer to graph H' as the *parent-graph* of H''. Note that the level of H' is either i or $(i-1)$. Assume that it is the latter. Then, from the algorithm's description, $|V(H'')| \leq \frac{2d(n)}{\beta} \cdot |V(H')|$ must hold.

We can now construct a *partitioning tree*, that contains a vertex $v(H')$ for every graph H' that was ever present in $\mathcal{H}$ over the course of the algorithm, an an edge between vertices $v(H')$ and $v(H'')$ whenever graph H' is a parent-graph of graph H''. The root of the tree is $v(H)$. Consider now again some graph H'', and the unique path P in the partitioning tree, connecting $v(H)$ to $v(H'')$. Denote the vertices on this path by $v(H) = v(H_0), v(H_1), \ldots, v(H_r) = v(H'')$, and assume that these vertices appear on path P in this order. For all $1 \leq i \leq r$, denote the level of graph H_i by L_i. Then $0 = L_1 \leq L_2 \leq \cdots \leq L_r$ must hold. Moreover, for every index $0 < i \leq r$, either $L_i = L_{i-1}$; or $L_i = L_{i-1} + 1$ hold. In the latter case, $|V(H_i)| \leq |V(H_{i-1})| \cdot \frac{2d(n)}{\beta}$. Denote $\Delta = \frac{\log n}{\log\left(\frac{\beta}{2d(n)}\right)}$. We claim that $L_r \leq \Delta$. Indeed, assume for contradiction that $L_r > \Delta$. Then there is a collection $J \subseteq \{1, \ldots, r\}$ of at least $\Delta + 1$ indices i, for which $L_i = L_{i-1} + 1$. But then:

$$|V(H'')| \leq n \cdot \left(\frac{2d(n)}{\beta}\right)^{\Delta+1} < 1,$$

a contradiction. We conclude that $L^* \leq \frac{\log n}{\log\left(\frac{\beta}{2d(n)}\right)}$. Substituting $\beta = 2^{8(\log n)^{1/2+\varepsilon}}$ and $d(n) \leq 2^{(\log n)^{\varepsilon}}$, we get that:

$$L^* \leq \frac{\log n}{\log\left(2^{6(\log n)^{1/2+\varepsilon}}\right)} \leq (\log n)^{1/2-\varepsilon}.$$

□

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:

$$|E^g \cup E^b| \geq \frac{|E(H)|}{\eta^{L^*}}.$$

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 3.11, $L^* \leq (\log n)^{1/2-\varepsilon}$, 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 $\text{poly}(n)$ iterations, and the running time of each iteration is bounded by $O(T(n) \cdot \text{poly}(n))$, so the total running time of the algorithm is at most $O(T(n) \cdot \text{poly}(n))$. □

We are now ready to complete the proof of Theorem 3.7. 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 2.1, 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 $O(T(N^2) \cdot \text{poly}(N))$. Denote $T'(N) = O(T(N^2) \cdot \text{poly}(N))$ 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 $O(T'(N))$. Indeed, if N is greater than a sufficiently large constant, then we can use Algorithm $\mathcal{A}$, to obtain a solution whose approximation factor is $\alpha'(N) \leq O\left(2^{(2\log N)^{\varepsilon}}\right) \leq 2^{(4\log N)^{\varepsilon}} \leq \alpha''(N)$. Otherwise, we can solve the problem exactly via exhaustive search.

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 $\text{size}(\mathcal{I}) \leq n$ 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 $\beta = 2^{8(\log n)^{1/2+\varepsilon}}$, 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.

Initially, $E^b = \emptyset$. 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 3.10 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 3.4, 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 from Corollary 3.10. Recall that Corollary 3.10 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 $O(T(\text{poly}(n)) \cdot \text{poly}(n))$.

In order to complete the proof of Theorem 3.7, it is now enough to prove Theorem 3.8, which we do next.

3.4 Proof of Theorem 3.8

The proof partially relies on ideas and techniques from [CKN21]. 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. Let $\mathcal{I} = (X, Y, A, \mathcal{C}, \{G_C\}_{C \in \mathcal{C}})$ be the input instance of Bipartite 2-CSP, with $\text{size}(\mathcal{I}) \leq n$. For convenience, we denote $H = H(\mathcal{I})$. If $|E(H)| \leq \beta^3$, then graph H is β^3-good, since we can compute an assignment to the variables of $X \cup Y$ that satisfies at least one constraint of $\mathcal{C}$. Therefore, we assume from now on that $|E(H)| > \beta^3$. We can also assume that graph H contains no isolated vertices, as isolated vertices of H correspond to variables that do not participate in any constraints, and can be discarded.

The proof consists of four steps. In the first step, in order to simplify the proof, we will regularize graph H, by computing a "nice" subgraph $\tilde{H} \subseteq H$. In the second step, we will define an *assignment graph* associated with $\tilde{H}$, and we will use it in order to obtain an instance of Bipartite Densest (k_1, k_2)-Subgraph, to which algorithm $\mathcal{A}$ will then be applied. In the last two steps, we will use the outcome of algorithm $\mathcal{A}$ in order to either correctly establish that graph H is β^3-good, or to compute a bad subset of constraints, or a subgraph H' of H as required. We now describe each of the three steps in turn.

3.4.1 Step 1: Regularization

In this step we will compute a subgraph $\tilde{H}$ of H that has a convenient structure. We refer to graphs with such structure as *nice subgraphs* of H, and define them next.

Definition 3.12 (Nice Subgraph of H)**.** *Let $\tilde{H} = (\tilde{X}, \tilde{Y}, \tilde{E})$ be a subgraph of H, and let $d_1, d_2 \geq 1$ be parameters. We say that $\tilde{H}$ is a (d_1, d_2)-nice subgraph of H, if the following hold:*

- *For every vertex $x \in \tilde{X}$, $d_1 \leq \deg_H(x) < 2d_1$;*
- *For every vertex $y \in \tilde{Y}$, $d_2 \leq \deg_H(y) < 8d_2 \log n$ and $d_2 \leq \deg_{\tilde{H}}(y) < 2d_2$; and*
- $|\tilde{E}| \geq \frac{d_1}{4 \log n} \cdot |\tilde{X}|$.

We say that $\tilde{H}$ is a nice *subgraph of H' if it is a (d_1, d_2)-nice subgraph of H for any pair $d_1, d_2 \geq 1$ of parameters.*

The first step of our algorithm is summarized in the following claim, that allows us to compute a nice subgraph $\tilde{H}$ of H that contains many edges of H.

Claim 3.13. *There is an algorithm with running time $O(\text{poly}(n))$, that computes parameters $d_1, d_2 > 1$, and a subgraph $\tilde{H}$ of H, such that $\tilde{H}$ is a (d_1, d_2)-nice subgraph of H, and $|E(\tilde{H})| \geq \frac{|E(H)|}{8 \log^2 n}$.*

Proof: The proof uses standard regularization techniques, and consists of three steps. Denote $H = (X, Y, E)$.

In the first step, we partition the vertices of X into groups $S_0, \ldots, S_q$, for $q = \lceil \log n \rceil$, where for all $0 \leq i \leq q$, $S_i = \{x \in X \mid 2^i \leq \deg_H(x) < 2^{i+1}\}$. We also partition the set E of edges into subsets $E_0, \ldots, E_q$, where for all $0 \leq i \leq q$, set E_i contains all edges $e \in E$ that are incident to vertices of S_i. Clearly, there is an index $0 \leq i^* \leq q$, with $|E_{i^*}| \geq \frac{|E(H)|}{2 \log n}$. We let $\tilde{X} = S_{i^*}$, and we let H_1 be the graph whose vertex set is $\tilde{X} \cup Y$, and edge set is E_{i^*}. We also define $d_1 = 2^{i^*}$. Clearly, for every vertex $x \in \tilde{X}$, $d_1 \leq \deg_H(x) < 2d_1$. This completes the first regularization step.

We now proceed to describe our second step, in which we consider the vertices of $y \in Y$ one by one. We say that a vertex $y \in Y$ is *bad*, if $\deg_{H_1}(y) < \frac{\deg_H(y)}{4 \log n}$. Let $Y'' \subseteq Y$ be the set of all bad vertices, and let $Y' = Y \setminus Y''$ be the set of all remaining vertices of Y, that we refer to as *good vertices*. We use the following observation.

Observation 3.14. $\sum_{y \in Y''} \deg_{H_1}(y) \leq \frac{|E(H_1)|}{2}$.

Proof: Since, for every bad vertex y, $\deg_{H_1}(y) < \frac{\deg_H(y)}{4 \log n}$, we get that:

$$\sum_{y \in Y''} \deg_{H_1}(y) < \sum_{y \in Y''} \frac{\deg_H(y)}{4 \log n} \leq \frac{|E(H)|}{4 \log n}.$$

Since, as observed above, $|E(H_1)| \geq \frac{|E(H)|}{2 \log n}$, the observation follows. □

We let H_2 be a graph that is obtained from H_1, by discarding the vertices of Y'' from it. Therefore, $V(H_2) = \tilde{X} \cup Y'$. Additionally, from Observation 3.14, $|E(H_2)| \geq \frac{|E(H_1)|}{2} \geq \frac{|E(H)|}{4 \log n}$.

Lastly, in our third step, we perform a geometric grouping of the vertices of Y' by their degree in H_2. Specifically, we let $r = \lceil \log n \rceil$, and we partition the vertices of Y' into sets $S'_0, \ldots, S'_r$, where for $0 \leq j \leq r$, $S'_j = \{y \in Y' \mid 2^j \leq \deg_{H_2}(y) < 2^{j+1}\}$. As before, we also partition the set $E(H_2)$ of edges into subsets $E'_0, \ldots, E'_r$, where for $0 \leq j \leq r$ set E'_j contains all edges $e \in E(H_2)$ that are incident to the vertices of S'_j. As before, there must be an index $0 \leq j^* \leq r$ with $|E'_{j^*}| \geq \frac{|E(H_2)|}{2 \log n} \geq \frac{|E(H)|}{8 \log^2 n}$. We set $\tilde{Y} = S'_{j^*}$, $d_2 = 2^{j^*}$, and we let $\tilde{H}$ be the graph whose vertex set is $\tilde{X} \cup \tilde{Y}$, and edge set is E'_{j^*}. We now verify that this graph has all required properties.

First, as observed already, for every vertex $x \in \tilde{X}$, $d_1 \leq \deg_H(x) < 2d_1$. Let $\tilde{E} = E(\tilde{H})$. As observed already, $|\tilde{E}| \geq \frac{|E(H)|}{8 \log^2 n}$. Moreover, since, for every vertex $x \in \tilde{X}$, $\deg_{H_1}(x) = \deg_H(x) \geq d_1$, we get that $|E(H_1)| \geq d_1 \cdot |\tilde{X}|$, and so $|\tilde{E}| \geq \frac{|E(H_2)|}{2 \log n} \geq \frac{|E(H_1)|}{4 \log n} \geq \frac{d_1}{4 \log n} \cdot |\tilde{X}|$.

Consider now some vertex $y \in \tilde{Y}$. From the definition of graph $\tilde{H}$, it is immediate to verify that $\deg_{\tilde{H}}(y) = \deg_{H_2}(y)$. Therefore, $d_2 \leq \deg_{\tilde{H}}(y) < 2d_2$. Clearly, $\deg_H(y) \geq \deg_{\tilde{H}}(y) \geq d_2$. Lastly, since vertex y is good, we get that:

$$\deg_{\tilde{H}}(y) = \deg_{H_2}(y) = \deg_{H_1}(y) \geq \frac{\deg_H(y)}{4 \log n}.$$

Since $\deg_{\tilde{H}}(y) < 2d_2$, we get that $\deg_H(y) \leq (4 \log n) \deg_{\tilde{H}}(y) < 8d_2 \log n$. We conclude that $d_2 \leq \deg_H(y) < 8d_2 \log n$. □

3.4.2 Step 2: Assignment Graph and Reduction to Bipartite Densest (k_1, k_2)-Subgraph

Recall that we have computed, in the first step, a subgraph $\tilde{H} = (\tilde{X}, \tilde{Y}, \tilde{E})$ of the graph $H = H(\mathcal{I})$. Since every edge of H is associated with a distinct constraint in $\mathcal{C}$, we can define a collection $\tilde{\mathcal{C}} \subseteq \mathcal{C}$ of constraints corresponding to the edges of $\tilde{H}$: $\tilde{\mathcal{C}} = \{C \in \mathcal{C} \mid e_C \in \tilde{E}\}$.

Next, we define a bipartite graph $G = (U, V, \hat{E})$, called *assignment graph*, that is associated with graph $\tilde{H}$. For every variable $z \in \tilde{X} \cup \tilde{Y}$, we define a set $R(z) = \{v(z, a) \mid 1 \leq a \leq A\}$ of vertices that represent the possible assignments to variable z. We then set $U = \bigcup_{x \in \tilde{X}} R(x)$, and $V = \bigcup_{y \in \tilde{Y}} R(y)$. The set of vertices of G is defined to be $U \cup V$.

In order to define the edges, consider any constraint $C = C(x, y) \in \tilde{\mathcal{C}}$. We define a set $E(C)$ of at most $d(n) \cdot A$ edges corresponding to C, as follows: we add an edge between vertex $v(x, a)$ and vertex $v(y, a')$ to $E(C)$ if assignments a to x and a' to y satisfy the constraint C. Since instance $\mathcal{I}$ is a $d(n)$-to-$d(n)$ instance of Bipartite 2-CSP, every vertex of $R(x) \cup R(y)$ is incident to at most $d(n)$ edges of $E(C)$. We then let $E(G) = \bigcup_{C \in \tilde{\mathcal{C}}} E(C)$.

Let $k_1 = |\tilde{X}|$ and $k_2 = |\tilde{Y}|$. We can then view graph G, together with parameters k_1 and k_2 as an instance of the Bipartite Densest (k_1, k_2)-Subgraph problem, $\mathrm{DkS}(G, k_1, k_2)$. Notice that $|V(G)| \leq |\tilde{\mathcal{C}}| \cdot A^2 \leq |\mathcal{C}| \cdot A^2 \leq \mathrm{size}(\mathcal{I}) \leq n$. We apply Algorithm $\mathcal{A}$ to instance $\mathrm{DkS}(G, k_1, k_2)$ of Bipartite Densest (k_1, k_2)-Subgraph, and we let S be the solution that the algorithm returns. Denote $G' = G[S]$, so the value of the solution is $|E(G')|$. Assume first that $|E(G')| < \frac{|\tilde{\mathcal{C}}|}{4\alpha(n)}$. We use the

following observation to show that, in this case, the set $\tilde{\mathcal{C}}$ of constraints is bad.

Observation 3.15. *If the set $\tilde{\mathcal{C}}$ of constraints is not bad, then the value of the optimal solution to instance* $\mathrm{DkS}(G, k_1, k_2)$ *of* Bipartite Densest (k_1, k_2)-Subgraph *is at least* $\frac{|\tilde{\mathcal{C}}|}{4}$.

Proof: Assume that the set $\tilde{\mathcal{C}}$ of constraints is not bad. Then there is an assignment f to variables of $\tilde{X} \cup \tilde{Y}$ that satisfies more than $\frac{|\tilde{\mathcal{C}}|}{4}$ constraints of $\tilde{\mathcal{C}}$. For each variable $z \in \tilde{X} \cup \tilde{Y}$, we denote the corresponding assignment by $f(z)$. Let S' be the set of vertices of G that contains, for every variable $x \in \tilde{X}$, vertex $v(x, f(x))$, and for every variable $y \in \tilde{Y}$, vertex $v(y, f(y))$. Then S' is a valid solution to instance $\mathrm{DkS}(G, k_1, k_2)$ of Bipartite Densest (k_1, k_2)-Subgraph. Moreover, for every constraint $C \in \tilde{\mathcal{C}}$ that is satisfied by the assignment f, an edge of $E(C)$ must be contained in $G[S']$. Therefore, the value of solution S' is at least $\frac{|\tilde{\mathcal{C}}|}{4}$. □

From Observation 3.15, if the set $\tilde{\mathcal{C}}$ of constraints is not bad, then algorithm $\mathcal{A}$ must have returned a solution whose value is at least $\frac{|\tilde{\mathcal{C}}|}{4\alpha(n)}$. Therefore, if the value of the solution S that the algorithm returns is less than $\frac{|\tilde{\mathcal{C}}|}{4\cdot\alpha(n)}$, then we terminate the algorithm, and return $\tilde{\mathcal{C}}$ as a bad set of constraints. Recall that, from Claim 3.13, $|\tilde{\mathcal{C}}| = |E(\tilde{H})| \geq \frac{|E(H)|}{8\log^2 n} = \frac{|\mathcal{C}|}{8\log^2 n}$. From now on we assume that the value of the soution S is at least $\frac{|\tilde{\mathcal{C}}|}{4\alpha(n)}$. We will use the set S of vertices of G, in order to either correctly certify that graph H is β^3-good, or to compute a subgraph $H' \subseteq H$ with the required properties. It will be convenient for us to further regularize graph G', which we do in the next step.

3.4.3 Step 3: Further Regularization

It would be convenient for us to further regularize graph G', by computing a subgraph $G'' \subseteq G'$, with $|E(G'')|$ roughly comparable to $|E(G')|$, that has the following additional properties. First, for all variables $x \in \tilde{X}$ with $R(x) \cap V(G'') \neq \emptyset$, the cardinalities of the sets $R(x) \cap V(G'')$ of vertices are roughly equal to each other (to within factor 2). Similarly, for all variables $y \in \tilde{Y}$ with $R(y) \cap V(G'') \neq \emptyset$, the cardinalities of the sets $R(y) \cap V(G'')$ of vertices are roughly equal to each other. Lastly, for all constraints $C \in \tilde{\mathcal{C}}$ with $E(C) \cap E(G'') \neq \emptyset$, the cardinalities of the sets $E(C) \cap E(G'')$ of edges are roughly equal to each other. In this step, we compute a subgraph $G'' \subseteq G'$ with all these properties. The algorithm is summarized in the following claim.

Claim 3.16. *There is an algorithm with running time $O(\mathrm{poly}(n))$, that computes a subgraph $G'' \subseteq G'$, subsets $X^* \subseteq \tilde{X}$, $Y^* \subseteq \tilde{Y}$ of variables, a subset $\mathcal{C}^* \subseteq \tilde{\mathcal{C}}$ of constraints, and integers $q, q', r \geq 1$, such that, if we denote, for every variable $z \in X \cup Y$, $R'(z) = R(z) \cap V(G'')$, and for every constraint $C \in \mathcal{C}$, $E'(C) = E(C) \cap E(G'')$, then the following hold:*

- *for every variable $x \in X^*$, $2^q \leq |R'(x)| < 2^{q+1}$, and for $x \notin X^*$, $R'(x) = \emptyset$;*
- *for every variable $y \in Y^*$, $2^{q'} \leq |R'(y)| < 2^{q'+1}$, and for $y \notin Y^*$, $R'(y) = \emptyset$;*
- *for every constraint $C \in \mathcal{C}^*$, $2^r \leq |E'(C)| < 2^{r+1}$, and for $C \notin \mathcal{C}^*$, $E'(C) = \emptyset$; and*
- $|\mathcal{C}^*| \geq \frac{|\tilde{\mathcal{C}}|}{2^{r+6}\cdot\alpha(n)\cdot\log^3 n}$.

Proof: The proof follows a standard regularization process. Let $E^0 = E(G')$, so that $|E^0| \geq \frac{|\tilde{\mathcal{C}}|}{4 \cdot \alpha(n)}$.

Our first step regularizes the variables of $\tilde{X}$. We group the variables of $\tilde{X}$ into sets $J_0, J_1, \ldots, J_{\lceil \log A \rceil}$. For all $0 \leq i \leq \lceil \log A \rceil$, we let $J_i = \{x \in \tilde{X} \mid 2^i \leq |R(x) \cap V(G')| < 2^{i+1}\}$. Note that, if $R(x) \cap V(G') = \emptyset$, then variable x does not belong to any of the groups that we have defined. We also partition the edges of E^0 into groups $E_0, E_1, \ldots, E_{\lceil \log A \rceil}$, where for all $0 \leq i \leq \lceil \log A \rceil$, group E_i contains all edges $e \in E^0$ that are incident to the vertices of $\bigcup_{x \in J_i} (R(x) \cap V(G'))$. It is easy to verify that $(E_0, \ldots, E_{\lceil \log A \rceil})$ is indeed a partition of the set E^0 of edges. Therefore, there is an index $0 \leq q \leq \lceil \log A \rceil$ with $|E_q| \geq \frac{|E^0|}{2 \log A} \geq \frac{|E^0|}{2 \log n}$. We then set $X^* = J_q$. For each such variable $x \in X^*$, we set $R'(x) = R(x) \cap V(G')$, and for each variable $x \in \tilde{X} \setminus X^*$, we set $R'(x) = \emptyset$. We also let $E^1 = E_q$. From the above discussion, $|E^1| \geq \frac{|E^0|}{2 \log n}$, and, for every variable $x \in X^*$, $2^q \leq |R'(x)| < 2^{q+1}$. Note that all edges of E^1 are incident to vertices of $\bigcup_{x \in X^*} R'(x)$.

Our second step is to regularize the variables of $\tilde{Y}$, exactly as before. We group the variables of $\tilde{Y}$ into sets $J'_0, J'_1, \ldots, J'_{\lceil \log A \rceil}$. For all $0 \leq i' \leq \lceil \log A \rceil$, we let $J'_{i'} = \{y \in \tilde{Y} \mid 2^{i'} \leq |R(x) \cap V(G')| < 2^{i'+1}\}$. As before, if $R(y) \cap V(G') = \emptyset$, then variable y does not belong to any of the sets that we have defined. We also partition the edges of E^1 into sets $E'_0, E'_1, \ldots, E'_{\lceil \log A \rceil}$, where for all $0 \leq i' \leq \lceil \log A \rceil$, set $E_{i'}$ contains all edges $e \in E^1$ that are incident to the vertices of $\bigcup_{y \in J_{i'}} (R(y) \cap V(G'))$. As before, $(E'_0, \ldots, E'_{\lceil \log A \rceil})$ is a partition of the set E^1 of edges. Therefore, there is an index $0 \leq q' \leq \lceil \log A \rceil$ with $|E'_{q'}| \geq \frac{|E^1|}{2 \log A} \geq \frac{|E^1|}{2 \log n} \geq \frac{|E^0|}{4 \log^2 n}$. We then set $Y^* = J'_{q'}$. For each variable $y \in Y^*$, we let $R'(y) = R(y) \cap V(G')$, and for each variable $y \in \tilde{Y} \setminus Y^*$, we set $R'(y) = \emptyset$. We also let $E^2 = E'_{q'}$. From the above discussion, $|E^2| \geq \frac{|E^0|}{4 \log^2 n}$, and, for every variable $y \in Y^*$, $2^{q'} \leq |R'(y)| < 2^{y+1}$. Notice that every edge of E^2 is incident to a vertex of $\bigcup_{x \in X^*} R'(x)$ and a vertex of $\bigcup_{y \in Y^*} R'(y)$.

Our third and final step is to regularize the constraints. Recall that for each constraint $C = C(x, y) \in \tilde{\mathcal{C}}$, we defined a set $E(C)$ of edges. Since $|R(x)| = A$ and $|R(y)| = A$, $|E(C)| \leq A^2 \leq n$ must hold. We group all constraints $C \in \tilde{\mathcal{C}}$ into sets $\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_{\lceil \log n \rceil}$, where for all $0 \leq j \leq \lceil \log n \rceil$, set $\mathcal{C}_j$ contains all constraints $C \in \tilde{\mathcal{C}}$ with $2^j \leq |E(C) \cap E^2| < 2^{j+1}$. Note that, if $E(C) \cap E^2 = \emptyset$, then constraint C does not belong to any set. Next, we define a partition $E''_0, E''_1, \ldots, E''_{\lceil \log n \rceil}$ of the set E^2 of edges: for $0 \leq j \leq \lceil \log n \rceil$, set E''_j contains all edges in $\bigcup_{C \in \mathcal{C}_j} (E(C) \cap E^2)$. It is easy to verify that $(E''_0, \ldots, E''_{\lceil \log n \rceil})$ is indeed a partition of E^2. Therefore, there must be an index $0 < r \leq \lceil \log n \rceil$, with $|E''_r| \geq \frac{|E^2|}{2 \log n} \geq \frac{|E^0|}{8 \log^3 n}$. We let $\mathcal{C}^* = \mathcal{C}_r$ and $E^3 = E''_r$. Since every constraint $C \in \mathcal{C}^*$ contributes at most 2^{r+1} edges to E^3, we get that:

$$|\mathcal{C}^*| \geq \frac{|E^3|}{2^{r+1}} \geq \frac{|E^0|}{2^{r+4} \log^3 n} \geq \frac{|\tilde{\mathcal{C}}|}{2^{r+6} \cdot \alpha(n) \cdot \log^3 n}.$$

For every constraint $C \in \mathcal{C}^*$, we let $E'(C) = E(C) \cap E^2$, and for every constraint $C \in \tilde{\mathcal{C}} \setminus \mathcal{C}^*$, we let $E'(C) = \emptyset$. We are now ready to define graph G''. Its vertex set is $\left(\bigcup_{x \in X^*} R'(X)\right) \cup \left(\bigcup_{y \in Y^*} R'(y)\right)$, and its edge set is $\bigcup_{C \in \mathcal{C}^*} E'(C) = E^3$. It is immediate to verify, from the above discussion, that graph G', sets X^*, Y^* of variables, and set $\mathcal{C}^*$ of constraints have all required properties. □

In the next observation, we establish some useful bounds on the cardinalities of the sets X^*, Y^* of variables, and the set $\mathcal{C}^*$ of constraints.

Observation 3.17. *All of the following bounds hold:*

- $|X^*| \leq \frac{|\tilde{X}|}{2^q}$;
- $|Y^*| \leq \frac{|\tilde{Y}|}{2^{q'}}$;
- $|\mathcal{C}^*| \leq 2d_1|X^*| \leq \frac{2d_1|\tilde{X}|}{2^q}$; *and*
- $|\mathcal{C}^*| \leq 2d_2|Y^*| \leq \frac{2d_2|\tilde{Y}|}{2^{q'}}$.

Proof: From the definition of the instance (G, k_1, k_2) of Bipartite Densest (k_1, k_2)-Subgraph, graph G' contains at most $k_1 = |\tilde{X}|$ vertices of $\bigcup_{x \in \tilde{X}} R(x)$, and at most $k_2 = |\tilde{Y}|$ vertices of $\bigcup_{y \in \tilde{Y}} R(y)$. Since, for every variable $x \in X^*$, $R'(x) \subseteq V(G')$ and $|R'(x)| \geq 2^q$, we get that $|X^*| \leq \frac{|\tilde{X}|}{2^q}$. Similarly, $|Y^*| \leq \frac{|\tilde{Y}|}{2^{q'}}$.

Recall that, since $\tilde{H}$ is a nice subgraph of H, for every vertex $x \in \tilde{H}$, $\deg_{\tilde{H}}(x) \leq \deg_H(x) \leq 2d_1$, and so x may participate in at most $2d_1$ constraints of $\tilde{\mathcal{C}}$. Since $\mathcal{C}^* \subseteq \tilde{\mathcal{C}}$ and $X^* \subseteq \tilde{X}$, every variable $x \in X^*$ may participate in at most $2d_1$ constraints of $\mathcal{C}^*$. Therefore, $|\mathcal{C}^*| \leq 2d_1|X^*|$.

Similarly, since $\tilde{H}$ is a nice subgraph of H, for every vertex $y \in \tilde{Y}$, $\deg_{\tilde{H}}(y) \leq 2d_2$. Using the same arguments as before, $|\mathcal{C}^*| \leq 2d_2|Y^*|$. □

Recall that, from Claim 3.16, $|\mathcal{C}^*| \geq \frac{|\tilde{\mathcal{C}}|}{2^{r+5} \cdot \alpha(n) \cdot \log^3 n}$. Since graph $\tilde{H}$ is a nice subgraph of H, we get that, for every vertex $y \in \tilde{Y}$, $\deg_{\tilde{H}}(y) \geq d_2$. Therefore, $|\tilde{\mathcal{C}}| \geq |\tilde{Y}| \cdot d_2$, and so:

$$|\mathcal{C}^*| \geq \frac{d_2 \cdot |\tilde{Y}|}{2^{r+6} \cdot \alpha(n) \cdot \log^3 n}. \tag{1}$$

Similarly, from the definition of a nice subgraph, $|\tilde{\mathcal{C}}| \geq \frac{d_1}{4 \log n} \cdot |\tilde{X}|$, and so:

$$|\mathcal{C}^*| \geq \frac{d_1 \cdot |\tilde{X}|}{2^{r+8} \cdot \alpha(n) \cdot \log^4 n}. \tag{2}$$

Lastly, we show that both $2^q, 2^{q'}$ are close to 2^r, in the following corollary of Observation 3.17.

Corollary 3.18. *The following inequalities hold:*

- $\frac{2^r}{2d(n)} \leq 2^q \leq 2^{r+8} \cdot \alpha(n) \cdot \log^4 n$; *and*
- $\frac{2^r}{2d(n)} \leq 2^{q'} \leq 2^{r+6} \cdot \alpha(n) \cdot \log^3 n$.

Proof: Consider some constraint $C = C(x, y) \in \mathcal{C}^*$, and recall that $|E'(C)| \geq 2^r$. From the definition of the d-to-d instances, every vertex $v(x, a) \in R(x)$ may be incident to at most $d(n)$ edges of $E(C)$. Since all edges of $E'(C)$ are incident to vertices of $R'(x)$, and $|R'(x)| \leq 2^{q+1}$, we get that $|E'(C)| \leq d(n) \cdot |R'(x)| \leq d(n) \cdot 2^{q+1}$. We conclude that $2^r \leq d(n) \cdot 2^{q+1}$.

Similarly, every vertex $v(y, a') \in R(y)$ may be incident to at most $d(n)$ edges of $E(C)$. Since all edges of $E'(C)$ are incident to vertices of $R'(y)$, and $|R'(y)| \leq 2^{q'+1}$, we get that $|E'(C)| \leq d(n) \cdot 2^{q'+1}$. This proves the inequalities $\frac{2^r}{2d(n)} \leq 2^q$ and $\frac{2^r}{2d(n)} \leq 2^{q'}$.

Next, we prove that $2^{q'} \leq 2^{r+6} \cdot \alpha(n) \cdot \log^3 n$. Recall that, from Inequality 1, $|\mathcal{C}^*| \geq \frac{d_2 \cdot |\tilde{Y}|}{2^{r+6} \cdot \alpha(n) \cdot \log^3 n}$. On the other hand, from the definition of a nice subgraph, every variable $y \in Y^*$ may participate in at most $2d_2$ constraints of $\mathcal{C}^*$, and, from Observation 3.17, $|Y^*| \leq \frac{|\tilde{Y}|}{2^{q'}}$. Therefore:

$$|\mathcal{C}^*| \leq 2d_2 \cdot |Y^*| \leq \frac{2d_2|\tilde{Y}|}{2^{q'}}. \tag{3}$$

Combining the two inequalities, we get that: $2^{q'} \leq 2^{r+6} \cdot \alpha(n) \cdot \log^3 n$.

Lastly, we prove that $2^q \leq 2^{r+8} \cdot \alpha(n) \cdot \log^4 n$. Recall that, from Inequality 2, $|\mathcal{C}^*| \geq \frac{d_1 \cdot |\tilde{X}|}{2^{r+7} \cdot \alpha(n) \cdot \log^4 n}$ holds. As before, from the definition of a nice subgraph, every variable $x \in X^*$ may participate in at most $2d_1$ constraints of $\mathcal{C}^*$, and, from Observation 3.17, $|X^*| \leq \frac{|\tilde{X}|}{2^q}$. Therefore:

$$|\mathcal{C}^*| \leq 2d_1 \cdot |X^*| \leq \frac{2d_1 \cdot |\tilde{X}|}{2^q}. \tag{4}$$

Combining the two inequalities together, we get that: $2^q \leq 2^{r+8} \cdot \alpha(n) \cdot \log^4 n$. □

3.4.4 Step 4: Certifying that H is a Good Graph or Computing a Subgraph of H

We consider two cases, depending on whether $2^r \leq \beta$ holds. We start by showing that, if $2^r \leq \beta$, then graph H is β^3-good.

Observation 3.19. *If $2^r \leq \beta$, then graph H is β^3-good.*

Proof: We show that there exists an assignment to variables of $X \cup Y$ that satisfies at least $|\mathcal{C}|/\beta^3$ constraints of $\mathcal{C}$. In order to do it, we show a randomized algorithm that computes assignments to variables of $X \cup Y$, such that the expected number of satisfied constraints is at least $|\mathcal{C}|/\beta^3$.

The assignments are computed as follows. Consider a variable $x \in X$. If $x \notin X^*$, then we assign to x an arbitrary value from $[A]$. Assume now that $x \in X^*$. Recall that we have defined a set $R'(x) \subseteq R(x)$ of vertices, whose cardinality is at most 2^{q+1}. Set $R'(x)$ of vertices naturally defines a collection $\hat{A}(x) = \{a \in [A] \mid v(x, a) \in R'(x)\}$ of assignments to variable x, with $|\hat{A}(x)| \leq 2^{q+1}$. We choose an assignment $a \in \hat{A}(x)$ uniformly at random, and assign value a to x.

Assignments to variables of Y are defined similarly. Consider a variable $y \in Y$. If $y \notin Y^*$, then we assign to y an arbitrary value from $[A]$. Assume now that $y \in Y^*$. Recall that we have defined a set $R'(y) \subseteq R(y)$ of vertices, whose cardinality is at most $2^{q'+1}$. Set $R'(y)$ of vertices naturally defines

a collection $\hat{A}(y) = \{a \in [A] \mid v(y,a) \in R'(y)\}$ of assignments to variable y, with $|\hat{A}(y)| \leq 2^{q'+1}$. We choose an assignment $a' \in \hat{A}(y)$ uniformly at random, and assign value a' to y.

Recall that we have computed, in Claim 3.16, a collection $\mathcal{C}^* \subseteq \mathcal{C}$ of constraints, with $|\mathcal{C}^*| \geq \frac{|\tilde{\mathcal{C}}|}{2^{r+6} \cdot \alpha(n) \cdot \log^3 n}$. Consider now any constraint $C = C(x,y) \in \mathcal{C}^*$, and recall that $x \in X^*, y \in Y^*$ must hold. Recall that graph G'' contains a collection $E'(C) \subseteq E(C)$ of edges, with $|E'(C)| \geq 2^r$. Consider now any such edge $e = (v(x,a), v(y,A))$. We say that edge e *wins* if x is assigned value a, and y is assigned value a'. The probability that edge e wins is at least $\frac{1}{2^{q+1} \cdot 2^{q'+1}}$. Notice that at most one edge of $E'(C)$ may win, and so the probability that any edge of $E'(C)$ wins is at least $\frac{|E'(C)|}{2^{q+1} \cdot 2^{q'+1}} \geq \frac{2^r}{2^{q+1} \cdot 2^{q'+1}}$. If any edge of $E'(C)$ wins, the constraint C is satisfied by the assignment that the algorithm chooses. Therefore, the probability that a constraint $C \in \mathcal{C}^*$ is satisfied is at least $\frac{2^r}{2^{q+1} \cdot 2^{q'+1}}$.

Overall, the expected number of constraints that are satisfied by the assignment is at least:

$$\frac{|\mathcal{C}^*| \cdot 2^r}{2^{q+1} \cdot 2^{q'+1}} \geq \frac{|\tilde{\mathcal{C}}|}{2^8 \cdot \alpha(n) \cdot 2^q \cdot 2^{q'} \cdot \log^3 n}$$

Recall that, from Claim 3.13, $|\tilde{\mathcal{C}}| = |E(\tilde{H})| \geq \frac{|E(H)|}{8 \log^2 n} = \frac{|\mathcal{C}|}{8 \log^2 n}$, and, from Corollary 3.18, $2^q \cdot 2^{q'} \leq 2^{2r+14} \cdot (\alpha(n))^2 \cdot \log^7 n \leq 2^{14} \cdot \beta^2 \cdot (\alpha(n))^2 \cdot \log^7 n$, since we have assumed that $2^r \leq \beta$. Therefore, the expected number of constraints of $\mathcal{C}$ that are satisfied by the assignment is at least:

$$\frac{|\tilde{\mathcal{C}}|}{2^8 \cdot \alpha(n) \cdot 2^q \cdot 2^{q'} \cdot \log^3 n} \geq \frac{|\mathcal{C}|}{2^{11} \cdot \alpha(n) \cdot 2^q \cdot 2^{q'} \cdot \log^5 n} \geq \frac{|\mathcal{C}|}{2^{25} \cdot \beta^2 \cdot (\alpha(n))^3 \cdot \log^{12} n} \geq \frac{|\mathcal{C}|}{\beta^3},$$

since $\beta \geq 2^{27} (\alpha(n))^3 \log^{12} n$.

We conclude that there is an assignment to the variables of $X \cup Y$ that satisfies at least $|\mathcal{C}|/\beta^3$ constraints of $\mathcal{C}$, and so graph H is β^3-good. □

If $2^r \leq \beta$, then we terminate the algorithm and report that graph H is β-good.

From now on we assume that $2^r > \beta$. In this case, we return a subgraph a subgraph $H' = (X', Y', E')$ of $H(\mathcal{I})$, that is defined as follows: $X' = X^*$, $Y' = Y^*$, and $E' = E^*$. We now verify that this graph has all required properties.

Recall that, from Observation 3.17, $|X^*| \leq \frac{|\tilde{X}|}{2^q} \leq \frac{|X|}{2^q}$, from Corollary 3.18, $2^q \geq \frac{2^r}{2d(n)}$, and, from our assumption, $2^r > \beta$. Therefore:

$$|X^*| \leq \frac{|X|}{2^q} \leq \frac{|X| \cdot 2d(n)}{2^r} \leq \frac{2d(n)}{\beta} \cdot |X|.$$

Similarly, from Observation 3.17, $|Y^*| \leq \frac{|\tilde{Y}|}{2^{q'}} \leq \frac{|Y|}{2^{q'}}$, from Corollary 3.18, $2^{q'} \geq \frac{2^r}{2d(n)}$, and, from our assumption, $2^r > \beta$. Therefore:

$$|Y^*| \leq \frac{|Y|}{2^{q'}} \leq \frac{|Y| \cdot 2d(n)}{2^r} \leq \frac{2d(n)}{\beta} \cdot |Y|.$$

It now only remains to show that $|E^*| \geq \frac{\mathrm{vol}_H(X^* \cup Y^*)}{256d(n)\cdot\alpha(n)\cdot\log^4 n}$.

Recall that, from Inequality 2, $|\mathcal{C}^*| \geq \frac{d_1\cdot|\tilde{X}|}{2^{r+8}\cdot\alpha(n)\cdot\log^4 n}$ holds. Since, from Corollary 3.18, $2^r \leq 2^q\cdot 2d(n)$, we get that $|E^*| = |\mathcal{C}^*| \geq \frac{d_1\cdot|\tilde{X}|}{2^{q+9}\cdot d(n)\cdot\alpha(n)\cdot\log^4 n}$. At the same time, from the definition of a nice graph, for every vertex $x \in X^*$, $\deg_H(x) \leq 2d_1$, so $\mathrm{vol}_H(X^*) \leq 2d_1 \cdot |X^*| \leq \frac{2d_1\cdot|\tilde{X}|}{2^q}$, since $|X^*| \leq \frac{|\tilde{X}|}{2^q}$ from Observation 3.17. Therefore, $|E'| \geq \frac{\mathrm{vol}_H(X^*)}{1024d(n)\cdot\alpha(n)\cdot\log^4 n}$.

Similarly, from Inequality 1, $|\mathcal{C}^*| \geq \frac{d_2\cdot|\tilde{Y}|}{2^{r+6}\cdot\alpha(n)\cdot\log^3 n}$. Since, from Corollary 3.18, $2^r \leq 2^{q'}\cdot 2d(n)$, we get that $|E'| = |\mathcal{C}^*| \geq \frac{d_2\cdot|\tilde{Y}|}{2^{q'+7}\cdot d(n)\cdot\alpha(n)\cdot\log^3 n}$. At the same time, from the definition of a nice graph, for every vertex $y \in Y^*$, $\deg_H(y) \leq 2d_2$, so $\mathrm{vol}_H(Y^*) \leq 2d_2 \cdot |Y^*| \leq \frac{2d_2\cdot|\tilde{Y}|}{2^{q'}}$, since $|Y^*| \leq \frac{|\tilde{Y}|}{2^{q'}}$ from Observation 3.17. Therefore, $|E^*| \geq \frac{\mathrm{vol}_H(Y^*)}{1024d(n)\cdot\alpha(n)\cdot\log^3 n}$.

Altogether, we get that $|E'| \geq \frac{\mathrm{vol}_H(X^*\cup Y^*)}{2048d(n)\cdot\alpha(n)\cdot\log^4 n}$.

Note that every step of the algorithm, except Step 2, has running time $O(\mathrm{poly}(n))$, while the running time of Step 2 is $O(T(n) + \mathrm{poly}(n))$. Therefore, the total running time of the algorithm is at most $O(T(n)\cdot\mathrm{poly}(n))$.

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

In this section we provide reductions from the Dense k-Coloring and (r,h)-Graph Partitioning problems to Densest k-Subgraph, by proving the following theorem.

Theorem 4.1. *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 $O(\alpha(N^2)\cdot\mathrm{poly}\log N)$-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 $O(\alpha(N^2)\cdot\mathrm{poly}\log N)$-approximate solution to this instance.*

We provide the proof of the first assertion of the theorem, by showing a reduction from Dense k-Coloring to Densest k-Subgraph. The proof of the second assertion is similar and is deferred to Section B of Appendix. 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.

4.1 An LP-Relaxation and Its Rounding

Let $\mathrm{DkC}(G,k)$ be an input instance of the Dense k-Coloring problem, and denote $|V(G)| = N$. We let $\mathcal{R}$ the collection of all subsets of $V(G)$ containing at most k vertices, that is: $\mathcal{R} = \{S \subseteq V(G) \mid |S| \leq k\}$. For every set $S \in \mathcal{R}$ of vertices, we denote by $m(S) = |E_G(S)|$. We consider the following LP-relaxation of the Dense k-Coloring problem, that has a variable x_S for every set $S \in \mathcal{R}$ of vertices.

$$
\begin{array}{lll}
\text{(LP-P)} & & \\
\max & \sum_{S\in\mathcal{R}} m(S)\cdot x_S & \\
\text{s.t.} & & \\
& \sum_{\substack{S\in\mathcal{R}:\\ v\in S}} x_S \leq 1 & \forall v\in V(G)\\
& \sum_{S\in\mathcal{R}} x_S \leq N/k & \\
& x_S \geq 0 & \forall S\in\mathcal{R}
\end{array}
$$

It is easy to verify that (LP-P) is an LP-relaxation of the Dense k-Coloring problem. Indeed, consider a solution $(S_1,\ldots,S_r)$ to the input instance $\mathrm{DkC}(G,k)$, where $r = N/k$. For all $1 \leq i \leq r$, we set $x_{S_i} = 1$, and for every other set $S \in \mathcal{R}$, we set $x_S = 0$. This provides a feasible solution to (LP-P), whose value is precisely $\sum_{i=1}^{r} |E_G(S_i)|$. We denote the value of the optimal solution to (LP-P) by $\mathrm{OPT}_{\text{LP-P}}$. From the above discussion, $\mathrm{OPT}_{\text{LP-P}} \geq \mathrm{OPT}_{\mathrm{DkC}}(G,k)$.

Note that the number of variables in (LP-P) is at least $\binom{N}{k}$, and so we cannot solve it directly. We will show below an efficient algorithm that provides an approximate solution to (LP-P), whose support is reasonably small. Before we do so, we provide an LP-rounding algorithm, by proving the following claim.

Claim 4.2. *There is an efficient randomized algorithm, whose input consists of an instance* $\mathrm{DkC}(G,k)$ *of the* Dense k-Coloring *problem with* $N = |V(G)|$*, such that* N *is greater than a large enough constant, and a solution* $\{x_S \mid S \in \mathcal{R}\}$ *to (LP-P), in which the number of variables* x_S *with* $x_S > 0$ *is bounded by* $\mathrm{poly}(N)$*, and* $\sum_{S\in\mathcal{R}} m(S)\cdot x_S \geq OPT_{\text{LP-P}}/\beta$*, for some parameter* $1 \leq \beta \leq N^3$*; the solution is given by only specifying values of variables* x_S *that are non-zero. The algorithm with high probability returns an integral solution* $(S_1,\ldots,S_{N/k})$ *to instance* $\mathrm{DkC}(G,k)$*, such that* $\sum_{i=1}^{N/k} |E_G(S_i)| \geq \frac{\mathrm{OPT}_{\mathrm{DkC}}(G,k)}{2000\beta\log^3 N}$.

Proof: We assume that we are given a solution $\{x_S \mid S \in \mathcal{R}\}$ to (LP-P), in which the number of variables x_S with $x_S > 0$ is bounded by $\mathrm{poly}(N)$. Denote $C = \sum_{S\in\mathcal{R}} m(S)\cdot x_S$, so that $C \geq \frac{\mathrm{OPT}_{\text{LP-P}}}{\beta} \geq \frac{\mathrm{OPT}_{\mathrm{DkC}}(G,k)}{\beta}$ holds. We denote by $\mathcal{R}' \subseteq \mathcal{R}$ the collection of all sets $S \in \mathcal{R}$ with $x_S > 0$. Assume first that there is any set $S \in \mathcal{R}'$ with $m(S) \geq \frac{C}{300\log^3 N}$. Then we can obtain the

desired solution $(S_1, \ldots, S_{N/k})$ as follows. We start with $S_1 = S$ and $S_2 = \cdots = S_{N/k} = \emptyset$. We then iteratively add vertices of $V(G) \setminus S$ to sets $S_1, \ldots, S_{N/k}$ arbitrarily, to ensure that the cardinality of each such set is exactly k. It is immediate to verify that $\sum_{i=1}^{N/k} |E_G(S_i)| \geq \frac{C}{300 \log^3 N} \geq \frac{\mathrm{OPT}_{\mathrm{DkC}}(G,k)}{300\beta \log^3 N}$. Therefore, we assume from now on that, for every set $S \in \mathcal{R}'$, $m(S) < \frac{C}{300 \log^3 N}$.

We construct another collection $\mathcal{R}'' \subseteq \mathcal{R}'$ of subsets of vertices of G as follows. For every set $S \in \mathcal{R}'$, we add S to $\mathcal{R}''$ independently, with probability x_S. Clearly, $\mathbf{E}\left[\sum_{S \in \mathcal{R}''} m(S)\right] = \sum_{S \in \mathcal{R}} m(S) \cdot x_S = C$.

We say that a bad event $\mathcal{E}_1$ happens if some vertex $v \in V(G)$ lies in more than $5 \log N$ sets in $\mathcal{R}''$. We say that a bad event $\mathcal{E}_2$ happens if $|\mathcal{R}''| > (5N \log N)/k$. We say that a bad event $\mathcal{E}_3$ happens if $\sum_{S \in \mathcal{R}''} m(S) < \frac{C}{8}$. Lastly, we say that a bad event $\mathcal{E}$ happens if either of the events $\mathcal{E}_1, \mathcal{E}_2$, or $\mathcal{E}_3$ happen. We start with the following simple observation.

Observation 4.3. $\mathbf{Pr}[\mathcal{E}] \leq \frac{2}{N^3}$.

Proof: For every set $S \in \mathcal{R}'$, let Y_S be the random variable whose value is 1 if $S \in \mathcal{R}''$ and 0 otherwise.

Consider any vertex $v \in V(G)$. Denote by Z_v the number of vertex sets in $\mathcal{R}''$ containing v. Clearly, $Z_v = \sum_{\substack{S \in \mathcal{R}': \\ v \in S}} Y_S$. Therefore:

$$\mathbf{E}[Z_v] = \mathbf{E}\left[\sum_{\substack{S \in \mathcal{R}': \\ v \in S}} Y_S\right] = \sum_{\substack{S \in \mathcal{R}': \\ v \in S}} x_S \leq 1,$$

where the last inequality follows from the constraints of (LP-P).

By applying the Chernoff Bound from Lemma 2.2, we get that $\mathbf{Pr}[Z_v > 5 \log N] \leq 1/N^5$. Using the union bound over all vertices $v \in V(G)$, we get that $\mathbf{Pr}[\mathcal{E}_1] \leq 1/N^4$.

Notice that $\mathbf{E}[|\mathcal{R}''|] = \mathbf{E}\left[\sum_{S \in \mathcal{R}} Y_S\right] = \sum_{S \in \mathcal{R}} x_S \leq N/k$ from the constraints of (LP-P). Bad Event $\mathcal{E}_2$ happens if $|\mathcal{R}''| = \sum_{S \in \mathcal{R}} Y_S > (5N \log N)/k$. By applying the Chernoff Bound from Lemma 2.2 to the variables of $\{Y_S \mid S \in \mathcal{R}'\}$, we get that $\mathbf{Pr}[\mathcal{E}_2] = \mathbf{Pr}\left[\sum_{S \in \mathcal{R}'} Y_S > (5N \log N)/k\right] \leq 1/N^5$.

Lastly, we bound the probability of Event $\mathcal{E}_3$. Recall that we have assumed that, for every set $S \in \mathcal{R}'$, $m(S) < \frac{C}{300 \log^3 N}$ holds. We partition the collection $\mathcal{R}'$ of vertex subsets into $\rho = \lceil 2 \log N \rceil$ collections $\mathcal{R}_0, \ldots, \mathcal{R}_{\rho-1}$, as follows. For all $0 \leq i < \rho$, we let $\mathcal{R}_i = \{S \in \mathcal{R}' \mid 2^i \leq m(S) < 2^{i+1}\}$. For all $0 \leq i < \rho$, we denote $C_i = \sum_{S \in \mathcal{R}_i} m(S) \cdot x_S$. Clearly, $\sum_{i=0}^{\rho-1} C_i = C$. We say that an index $0 \leq i < \rho$ is *bad*, if $C_i < \frac{C}{8 \log N}$, and otherwise we say that i is a *good index*. Let $I^g, I^b \subseteq \{0, \ldots, \rho-1\}$ be the collections of good and bad indices, respectively. Since $\rho \leq 4 \log N$, we get that:

$$\sum_{i \in I^b} C_i \leq (4 \log N) \cdot \frac{C}{8 \log N} \leq \frac{C}{2}.$$

Therefore, $\sum_{i \in I^g} C_i \geq \frac{C}{2}$ holds. Consider now a good index $i \in I^g$. Recall that, for every set $S \in \mathcal{R}'$, $m(S) \leq \frac{C}{300 \log^3 N}$ holds, and so $2^i \leq \frac{C}{300 \log^3 N}$ must hold. Moreover, since $\sum_{S \in \mathcal{R}_i} m(S) \cdot x_S = C_i \geq \frac{C}{8 \log N}$, we get that:

$$\frac{C}{8\log N} \leq \sum_{S\in\mathcal{R}_i} m(S)x_S \leq 2^{i+1}\cdot\sum_{S\in\mathcal{R}_i} x_S \leq \frac{C}{150\log^3 N}\cdot\sum_{S\in\mathcal{R}_i} x_S.$$

We conclude that $\sum_{S\in\mathcal{R}_i} x_S \geq 16\log^2 N$ holds for every good index $i\in I^g$.

For a good index $i\in I^g$, we denote by $\mathcal{R}'_i=\mathcal{R}_i\cap\mathcal{R}''$, and we let $\tilde{\mathcal{E}}_i$ be the bad event that $|\mathcal{R}'_i|<\frac{\sum_{S\in\mathcal{R}_i}x_S}{2}$. From the Chernoff Bound (Lemma 2.2), and the fact that $\sum_{S\in\mathcal{R}_i}x_S\geq 16\log^2 N$, we get that $\mathbf{Pr}\left[\tilde{\mathcal{E}}_i\right]\leq e^{-2\log^2 N}<N^{-4}$, if N is sufficiently large. By applying the Union Bound to all indices $i\in I^g$, we get that the probability that any of the events in $\{\tilde{\mathcal{E}}_i\mid i\in I^g\}$ happens is bounded by $1/N^3$. Note that, if neither of the events in $\{\tilde{\mathcal{E}}_i\mid i\in I^g\}$ happen, then:

$$\begin{aligned}\sum_{S\in\mathcal{R}''} m(S) &\geq \sum_{i\in I^g} 2^i\cdot|\mathcal{R}'_i|\\ &\geq \sum_{i\in I^g} 2^i\cdot\frac{\sum_{S\in\mathcal{R}_i}x_S}{2}\\ &\geq \sum_{i\in I^g}\sum_{S\in\mathcal{R}_i}\frac{m(S)\cdot x_S}{4}\\ &\geq \sum_{i\in I^g}\frac{C_i}{4}\\ &\geq \frac{C}{8}.\end{aligned}$$

Therefore, if neither of the events in $\{\tilde{\mathcal{E}}_i\mid i\in I^g\}$ happen, then Event $\mathcal{E}_3$ also does not happen. We conclude that $\mathbf{Pr}\left[\mathcal{E}_3\right]\leq 1/N^3$.

Finally, from the Union Bound, we get that:

$$\mathbf{Pr}\left[\mathcal{E}\right]\leq\mathbf{Pr}\left[\mathcal{E}_1\right]+\mathbf{Pr}\left[\mathcal{E}_2\right]+\mathbf{Pr}\left[\mathcal{E}_3\right]\leq\frac{1}{N^4}+\frac{1}{N^5}+\frac{1}{N^3}\leq\frac{2}{N^3}.$$

□

Observe that we can efficiently check whether Event $\mathcal{E}$ happened. If Event $\mathcal{E}$ happens, then we terminate the algorithm with a FAIL. We assume from now on that Event $\mathcal{E}$ did not happen. In this case, we are guaranteed that $\sum_{S\in\mathcal{R}''}m(S)\geq\frac{C}{8}\geq\frac{\mathrm{OPT}_{\mathrm{DkC}}(G,k)}{8\beta}$. We denote $\mathcal{R}''=\{S_1,S_2,\ldots,S_z\}$, where the sets are indexed according to their value $m(S)$, so that $m(S_1)\geq m(S_2)\geq\cdots\geq m(S_z)$. We then let $\mathcal{S}=\{S_1,\ldots,S_{N/k}\}$ (if $z<N/k$, then we set $S_{z+1}=\cdots=S_{N/k}=\emptyset$). For all $1\leq i\leq N/k$, we denote $E_i=E_G(S_i)$, so $|E_i|=m(S_i)$, and we denote $E'=\bigcup_{i=1}^{N/k}E_i$. Recall that, since Event $\mathcal{E}$ did not happen, $|\mathcal{R}''|\leq(5N\log N)/k$ holds. Therefore:

$$|E'|\geq\frac{\sum_{S\in\mathcal{R}''}m(S)}{5\log N}\geq\frac{\mathrm{OPT}_{\mathrm{DkC}}(G,k)}{40\beta\log N}.$$

Note that the vertex sets in the family $\mathcal{S}$ may not be mutually disjoint. However, since Event $\mathcal{E}$ did not happen, every vertex of $V(G)$ may lie in at most $5\log N$ such sets. We now construct a new collection $\mathcal{S}' = \{S'_1, \ldots, S'_{N/k}\}$ of sets of vertices, as follows. Consider any vertex $v \in V(G)$, and let $S_{i_1}, S_{i_2}, \ldots, S_{i_a} \in \mathcal{S}$ be the sets of $\mathcal{S}$ containing v. Vertex v chooses an index $i^* \in \{i_1, \ldots, i_a\}$ at random, and is then added to S'_{i^*}.

Note that for all $1 \leq j \leq N/k$, for every vertex $v \in S_j$, the probability that $v \in S'_j$ is at least $1/(5\log N)$. We say that an edge $e = (u, v) \in E_j$ *survives* if both $u, v \in S'_j$. We denote by $E'' \subseteq E'$ the set of all edges that survive. Since $\mathbf{Pr}\left[u \in S'_j\right] \geq 1/(5\log N)$, $\mathbf{Pr}\left[v \in S'_j\right] \geq 1/(5\log N)$, and the two events are independent, we get that the probability that edge e survives is at least $1/(25\log^2 N)$. Overall, we get that:

$$\mathbf{E}\left[|E''|\right] \geq \frac{|E'|}{25\log^2 N} \geq \frac{\mathrm{OPT}_{\mathrm{DkC}}(G,k)}{1000\beta\log^3 N}.$$

We obtain a final solution $\mathcal{S}^*$ to instance $\mathrm{DkC}(G,k)$ of the Dense k-Coloring problem by starting with $\mathcal{S}^* = \mathcal{S}'$, and then partitioning the vertices of $V(G) \setminus \left(\bigcup_{j=1}^{N/k} S'_j\right)$ among the sets of $\mathcal{S}^*$ arbitrarily, until each such set contains exactly k vertices. Clearly, the value of the resulting solution is at least $|E''|$.

So far we have obtained a randomized algorithm that either returns FAIL (with probability at most $2/N^3$), or it returns a solution to instance $\mathrm{DkC}(G,k)$ of the Dense k-Coloring problem, whose expected value is at least $\frac{\mathrm{OPT}_{\mathrm{DkC}}(G,k)}{1000\beta\log^3 N}$.

Let p' be the probability that the algorithm returned a solution of value at least $\frac{\mathrm{OPT}_{\mathrm{DkC}}(G,k)}{3000\beta\log^3 N}$, given that it did not return FAIL. Note that the expected solution value, assuming the algorithm did not return FAIL, is at most $\frac{\mathrm{OPT}_{\mathrm{DkC}}(G,k)}{2000\beta\log^3 N} + p' \cdot \mathrm{OPT}_{\mathrm{DkC}}(G,k)$. Since this expectation is also at least $\frac{\mathrm{OPT}_{\mathrm{DkC}}(G,k)}{1000\beta\log^3 N}$, we get that $p' \geq \frac{1}{2000\beta\log^3 N}$. Overall, the probability that our algorithm successfully returns a solution of value at least $\frac{\mathrm{OPT}_{\mathrm{DkC}}(G,k)}{2000\beta\log^3 N}$ is $p' \cdot \mathbf{Pr}\left[\neg\mathcal{E}\right] \geq \Omega\left(\frac{1}{\beta\log^3 N}\right)$. By repeating the algorithm $\mathrm{poly}(N)$ times we can ensure that it successfully computes a solution of value at least $\frac{\mathrm{OPT}_{\mathrm{DkC}}(G,k)}{2000\beta\log^3 N}$ with high probability. □

4.2 Approximately Solving the LP-Relaxation

As observed already, (LP-P) has $\binom{N}{k}$ variables, and so we cannot solve it directly. Instead, we will use the Ellipsoids algorithm with an approximate separation oracle to its dual LP, that appears below. This LP has a variable z, and, additionally, for every vertex $v \in V(G)$, it has a variable y_v.

$$
\begin{array}{lll}
\text{(LP-D)} & & \\
\min & \frac{N}{k} \cdot z + \sum_{v \in V(G)} y_v & \\
\text{s.t.} & & \\
& z + \sum_{v \in S} y_v \geq m(S) & \forall S \in \mathcal{R} \\
& z \geq 0 & \\
& y_v \geq 0 & \forall v \in V(G)
\end{array}
$$

We denote the value of the optimal solution to (LP-D) by $\mathrm{OPT}_{\text{LP-D}}$.

Next, we define an approximate separation oracle, and provide such a separation oracle to (LP-D).

Approximate Separation Oracle. Consider the following general minimization Linear Program, whose variables are $\{x_1, \ldots, x_n\}$.

$$
\begin{array}{lll}
(P) & & \\
\min & \sum_{i=1}^{n} c_i x_i & \\
\text{s.t.} & & \\
& \sum_{i=1}^{n} A_{j,i} x_i \geq b_j & \forall 1 \leq j \leq m \\
& x_i \geq 0 & \forall 1 \leq i \leq n
\end{array}
$$

Let $\beta : \mathbb{Z}^+ \to \mathbb{Z}^+$ be an increasing function. A *randomized $\beta(n)$-approximate separation oracle* for (P) is an efficient randomized algorithm (that is, the running time of the algorithm is bounded by a polynomial function of its input size). The input to the algorithm is a set $\{x_i\}_{i=1}^{n}$ of non-negative real values. The algorithm either returns "accept", or it returns an LP-constraint (called a *violated constraint*) that does not hold for the given values $x_1, \ldots, x_n$. We say that the algorithm *errs* if it returns "accept" and yet there is some index $1 \leq j \leq m$ for which $\sum_{i=1}^{n} A_{j,i} x_i < b_j / \beta(n)$ holds. We require that the probability that the algorithm errs is at most $2/3$.

For the case where a linear program has a very large number of constraints, or the constraints are not given explicitly, one can use an approximate separation oracle, combined with the Ellipsoids algorithm, in order to compute an approximate LP-solution in time polynomial in the number of variables of the LP, provided that there is an Ellipsoid containing the feasible region, whose volume is not too large. For the case where a linear program has a large number of variables, but a relatively small number of constraints, we can use an approximate separation oracle for its dual LP in order to solve the original LP approximately. We start by providing an approximate separation oracle for (LP-D). We then show that this separation oracle can be used in order to obtain an approximate solution to (LP-P) in time poly(N).

Lemma 4.4. *Assume that there is an efficient $\alpha(n)$-approximation algorithm for the* Densest k-Subgraph *problem, where α is an increasing function, and n is the number of vertices in the input graph. Then there is a randomized $\beta(N)$-approximate separation oracle for (LP-D), where N is the number of variables in the input graph G, and $\beta(N) = O(\alpha(N^2) \cdot \log^2 N)$.*

Proof: Recall that we are given as input real values z and $\{y_v \mid v \in V(G)\}$. Clearly, we can efficiently check whether $z \geq 0$, and whether $y_v \geq 0$ for all $v \in V(G)$. If this is not the case, we can return the corresponding violated constraint.

We say that a set $S \in \mathcal{R}$ of vertices is *bad* if $z + \sum_{v \in S} y_v < m(S)/\beta$ holds, where $\beta = c \cdot \alpha(N^2) \cdot \log^2 N$, and c is a large enough constant whose value we set later. Our goal is to design an efficient algorithm that either returns a violated constraint of the LP (that is, a set $S \in \mathcal{R}$ of vertices for which $z + \sum_{v \in S} y_v < m(S)$ holds); or it returns "accept". We require that, if there exists a bad set S of vertices, then the probability that the algorithm returns "accept" is at most $2/3$.

It will be convenient for us to slightly modify the input values in $\{y_v \mid v \in V(G)\}$, as follows. We let m be the smallest integral power of 2 that is greater than $|E(G)|$. First, for every vertex $v \in V(G)$ with $y_v > m$, we let $y'_v = m$, and for every vertex $v \in V(G)$ with $y_v < 1/4$, we set $y'_v = 0$. For each remaining vertex v, we let y'_v be the smallest integral power of 2 that is greater than $4y_v$. Note that for every vertex v with $y'_v \neq 0$, $1 \leq y'_v \leq 4m$ holds, and y'_v is an integral power of 2. We also set $z' = 2z$. We say that a set $S \in \mathcal{R}$ of vertices is *problematic* if $z' + \sum_{v \in S} y'_v < 8m(S)/\beta$ holds. We need the following two observations regarding the new values $\{y'_v \mid v \in V(G)\}$.

Observation 4.5. *If $S \in \mathcal{R}$ is a bad set of vertices, then it is a problematic set of vertices.*

Proof: Recall that, if S is a bad set of vertices, then $z + \sum_{v \in S} y_v < m(S)/\beta$ must hold. Since, for every vertex $v \in V(G)$, $y'_v \leq 8y_v$ holds, and $z' = 2z$, we get that:

$$z' + \sum_{v \in S} y'_v \leq 2z + 8 \sum_{v \in S} y_v \leq 8 \left(z + \sum_{v \in S} y_v \right) < 8m(S)/\beta.$$

Therefore, set S is problematic. □

Observation 4.6. *Assume that there exists a set $S \in \mathcal{R}$ of vertices, for which $z' + \sum_{v \in S} y'_v < m(S)$ holds. Let $S' \subseteq S$ be the set of vertices obtained from S by deleting every vertex $v \in S$ that has no neighbors in S. Then $z + \sum_{v \in S'} y_v < m(S')$ holds.*

Proof: Since, for every vertex $v \in S \setminus S'$, no neighbor of v lies in S, we get that $m(S') = |E_G(S')| = |E_G(S)| = m(S)$. We partition the vertices of S' into two subsets: set X containing all vertices $v \in S'$ with $y_v < 1/4$, and set Y containing all remaining vertices. Clearly, $\sum_{v \in X} y_v < \frac{|X|}{4} \leq \frac{m(S')}{2}$ (since $m(S') \geq |S'|/2 \geq |X|/2$, as graph $G[S']$ contains no isolated vertices).

Assume for contradiction that $z + \sum_{v \in S'} y_v \geq m(S')$. Then:

$$z + \sum_{v \in Y} y_v \geq m(S') - \sum_{v \in X} y_v \geq m(S')/2.$$

We now consider two cases. The first case is when there is some vertex $v \in Y$ with $y_v \geq m$. In this case, $y'_v \geq m$ holds, and $z' + \sum_{v \in S} y'_v \geq m > m(S)$ holds, a contradiction.

Otherwise, for every vertex $v \in Y$, $y'_v \geq 4y_v$ holds. Since $z' = 2z$ also holds, we get that:

$$z' + \sum_{v \in S} y'_v \geq z' + \sum_{v \in Y} y'_v \geq 2z + 4 \sum_{v \in Y} y_v \geq m(S') = m(S),$$

a contradiction. □

From now on we focus on values $z', \{y'_v \mid v \in V(G)\}$. It is now enough to design an efficient randomized algorithm, that either computes a set $S \in \mathcal{R}$ of vertices, for which $z' + \sum_{v \in S} y'_v < m(S)$ holds, or returns "accept". It is enough to ensure that, if there is a problematic set $S \in \mathcal{R}$ of vertices, then the algorithm returns "accept" with probability at most 2/3. Indeed, if there is a bad set $S \in \mathcal{R}$ of vertices, then, from Observation 4.5, there is a problematic set of vertices, and the algorithm will return "accept" with probability at most 2/3. On the other hand, if the algorithm computes a set $S \in \mathcal{R}$ of vertices, for which $z' + \sum_{v \in S} y'_v < m(S)$ holds, then we can return the set $S' \subseteq S$ of vertices from the statement of Observation 4.6, that defines a violated constraint with respect to the original LP-values.

Our algorithm computes a random partition (A, B) of the vertices of G, where every vertex $v \in V(G)$ is independently added to A or to B with probability 1/2 each. Let $q = \log(8m)$. For all $1 \leq i \leq q$, we define a set $A_i \subseteq A$ of vertices: $A_i = \{v \in A \mid y'_v = 2^{i-1}\}$, and we let $A_0 = \{v \in A \mid y'_v = 0\}$. Clearly, $(A_0, \ldots, A_q)$ is a partition of the set A of vertices.

We compute a partition $(B_0, \ldots, B_q)$ of the vertices of B similarly. For all $0 \leq i, j \leq q$, we denote by $E_{i,j}$ the set of all edges $e = (u, v)$ with $u \in A_i$ and $v \in B_j$, and we define a bipartite graph $G_{i,j}$, whose vertex set is $A_i \cup B_j$, and edge set is $E_{i,j}$.

Recall that we have assumed 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. From Lemma 2.1, there exists an efficient $O(\alpha(\hat{n}^2))$-approximation algorithm for the Bipartite Densest (k_1, k_2)-Subgraph problem, where $\hat{n}$ is the number of vertices in the input graph. We denote this algorithm by $\mathcal{A}'$.

For every pair $0 \leq i, j \leq q$ of integers, and every pair $k_1, k_2 \geq 0$ of integers with $k_1 + k_2 \leq k$, we apply Algorithm $\mathcal{A}'$ for the Bipartite Densest (k_1, k_2)-Subgraph problem to graph $G_{i,j}$, with parameters k_1 and k_2. Let $S_{i,j}^{k_1,k_2}$ be the output of this algorithm, and let $m_{i,j}^{k_1,k_2}$ be the number of edges in the subgraph of $G_{i,j}$ that is induced by the set $S_{i,j}^{k_1,k_2}$ of vertices. We say that the application of algorithm $\mathcal{A}'$ is *successful* if $z' + \sum_{v \in S_{i,j}^{k_1,k_2}} y'_v < m_{i,j}^{k_1,k_2}$, and otherwise it is *unsuccessful*. If, for any quadruple (i, j, k_1, k_2) of indices, the application of algorithm $\mathcal{A}'$ was successful, then we return the resulting set $S = S_{i,j}^{k_1,k_2}$ of vertices. Clearly, $|S| \leq k_1 + k_2 \leq k$, so $S \in \mathcal{R}$ holds. Moreover, we are guaranteed that $z' + \sum_{v \in S} y'_v < m_{i,j}^{k_1,k_2} < m(S)$, as required. If every application of algorithm $\mathcal{A}'$ is unsuccessful, then we return "accept". The following observation will finish the proof of Lemma 4.4.

Observation 4.7. *Suppose there is a problematic set $S \in \mathcal{R}$ of vertices. Then the probability that the algorithm returns "accept" is at most 2/3.*

Proof: Let $S \in \mathcal{R}$ be a problematic set of vertices, so $z' + \sum_{v \in S} y'_v < 8m(S)/\beta$ holds. Let $E' = E_G[S]$, so $|E'| = m(S)$. Denote $A_S = A \cap S, B_S = B \cap S$, and let $E'' \subseteq E'$ be the set of edges e, such that exactly one endpoint of e lies in A. Clearly, for every edge $e \in E'$, $\mathbf{Pr}[e \in E''] = 1/2$.

Therefore, $\mathbf{E}\left[|E''|\right] = |E'|/2$. Let $\mathcal{E}'$ be the bad event that $|E''| < |E'|/8$, and let $p = \mathbf{Pr}\left[\mathcal{E}'\right]$. Clearly:

$$\mathbf{E}\left[|E''|\right] \leq p \cdot \frac{|E'|}{8} + (1-p) \cdot |E'| = |E'| \left(1 - \frac{7p}{8}\right).$$

Since $\mathbf{E}\left[|E''|\right] = |E'|/2$, we get that $p \leq 2/3$. Next, we show that, if Event $\mathcal{E}'$ does not happen, then the algorithm does not return "accept".

From now on we assume that Event $\mathcal{E}'$ did not happen, so $|E''| \geq |E'|/8$. Therefore:

$$z' + \sum_{v \in S} y'_v < \frac{8m(S)}{\beta} \leq \frac{64|E''|}{\beta}$$

holds.

Clearly, there must be a pair $0 \leq i, j \leq q$ of indices, such that $|E'' \cap E_{i,j}| \geq \frac{|E''|}{4q^2} \geq \frac{|E''|}{128 \log^2 m}$. We now fix this pair i, j of indices, and denote $A'_i = A_i \cap S$ and $B'_j = B_j \cap S$. We also denote $k_1 = |A'_i|$ and let $k_2 = |B'_j|$. Clearly, $k_1 + k_2 \leq k$ holds. Denote $M_{i,j} = |E'' \cap E_{i,j}|$. From our choice of indices i, j, we get that:

$$z' + \sum_{v \in A'_i \cup B'_j} y'_v \leq z' + \sum_{v \in S} y'_v \leq \frac{64|E''|}{\beta} \leq \frac{M_{i,j}}{\beta} \cdot (2^{13} \log^2 m).$$

Notice that the set $S' = A'_i \cup B'_j$ of vertices provides a solution to the instance of the Bipartite Densest (k_1, k_2)-Subgraph problem on graph $G_{i,j}$ with parameters k_1, k_2, whose value is at least $M_{i,j}$. Let $S'' = S_{i,j}^{k_1,k_2}$ be the set of vertices obtained by applying Algorithm $\mathcal{A}'$ to graph $G_{i,j}$ with parameters k_1, k_2. Since $|V(G_{i,j})| \leq N$, and since $\mathcal{A}'$ is an $O(\alpha(N^2))$-approximation algorithm for Bipartite Densest (k_1, k_2)-Subgraph, we are guaranteed that $|E_G(S'')| \geq \Omega\left(\frac{M_{i,j}}{\alpha(N^2)}\right)$. Recall that $|A \cap S''| \leq k_1$; $A \cap S'' \subseteq A_i$, and all vertices $v \in A_i$ have an identical value y'_v. Therefore, $\sum_{v \in A \cap S''} y'_v \leq \sum_{v \in A'_i} y'_v$. Using a similar reasoning, $\sum_{v \in B \cap S''} y'_v \leq \sum_{v \in B'_j} y'_v$. Overall, we then get that:

$$\begin{aligned} z' + \sum_{v \in S''} y'_v &\leq z' + \sum_{v \in S'} y'_v \\ &\leq \frac{M_{i,j}}{\beta} \cdot (2^{13} \cdot \log^2 m) \\ &\leq O\left(\frac{|E_G(S'')| \cdot \alpha(N^2) \cdot 2^{13} \cdot \log^2 m}{\beta}\right) \end{aligned}$$

Recall that $\beta = c \cdot \alpha(N^2) \cdot \log^2 N$. By letting the value of the constant c be high enough, we can ensure that $z' + \sum_{v \in S''} y'_v < |E_G(S'')|$, and so the application of algorithm $\mathcal{A}'$ to graph $G_{i,j}$ with parameters k_1 and k_2 is guaranteed to be successful. Therefore, if Event $\mathcal{E}'$ does not happen, and we set c to be a large enough constant, then our algorithm does not return "accept". Since $\mathbf{Pr}\left[\mathcal{E}'\right] \leq 2/3$, the observation follows. □ □

Approximately Solving (LP-P). We use standard methods for solving (LP-P) using approximate separation oracle for (LP-D).

For a collection $\mathcal{R}' \subseteq \mathcal{R}$ of vertex subsets, we define a linear program (LP($\mathcal{R}'$)), which is obtained from (LP-D) by only including the constraints associated with the subsets in $\mathcal{R}'$:

$$\begin{array}{lll} (\mathrm{LP}(\mathcal{R}')) & & \\ \min & \frac{N}{k} \cdot z + \sum_{v \in V(G)} y_v & \\ \text{s.t.} & & \\ & z + \sum_{v \in S} y_v \geq m(S) & \forall S \in \mathcal{R}' \\ & z \geq 0 & \\ & y_v \geq 0 & \forall v \in V(G) \end{array}$$

We denote by $\mathrm{OPT}(\mathcal{R}')$ the value of the optimal solution to (LP($\mathcal{R}'$)). Since a solution to (LP-D) defines a solution to (LP($\mathcal{R}'$)), we get that $\mathrm{OPT}(\mathcal{R}') \leq \mathrm{OPT}_{\text{LP-D}}$ We use the following lemma that allows us to compute a small collection $\mathcal{R}' \subseteq \mathcal{R}$ of vertex subsets, such that $\mathrm{OPT}(\mathcal{R}')$ is within a factor $\beta(N) = O(\alpha(N^2) \cdot \log^2 N)$ of $\mathrm{OPT}_{\text{LP-D}}$.

Claim 4.8. *Assume that there is an efficient $\alpha(n)$-approximation algorithm for* Densest k-Subgraph*, where α is an increasing function of n, and n is the number of vertices in the input graph. Then there is a randomized algorithm with running time $O(\mathrm{poly}(N))$, that computes a collection $\mathcal{R}' \subseteq \mathcal{R}$ of subsets of vertices with $|\mathcal{R}'| \leq O(\mathrm{poly}(N))$, such that, with high probability, $OPT(\mathcal{R}') \geq \Omega\left(OPT_{\text{LP-D}}/\beta(N)\right)$, where $\beta(N) = O(\alpha(N^2) \cdot \log^2 N)$.*

We prove Claim 4.8 below, after we provide an algorithm for approximately solving (LP-P) using it.

Recall that for every set $S \in \mathcal{R}$ of vertices, there is a variable x_S in the primal LP, (LP-P). We now consider the dual LP to LP($\mathcal{R}'$), that is defined as follows:

$$\begin{array}{lll} (\text{LP-P2}) & & \\ \max & \sum_{S \in \mathcal{R}'} m(S) \cdot x_S & \\ \text{s.t.} & & \\ & \sum_{\substack{S \in \mathcal{R}': \\ v \in S}} x_S \leq 1 & \forall v \in V(G) \\ & \sum_{S \in \mathcal{R}'} x_S \leq N/k & \\ & x_S \geq 0 & \forall S \in \mathcal{R}' \end{array}$$

Notice that this Linear Program can be obtained from (LP-P) by eliminating all variables x_S for $S \in \mathcal{R} \setminus \mathcal{R}'$. Since $|\mathcal{R}'| \leq \mathrm{poly}(N)$, this new linear program has at most $O(\mathrm{poly}(N))$ variables, and it has at most $\mathrm{poly}(N)$ constraints. Therefore, we can solve it in time $O(\mathrm{poly}(N))$ using standard algorithms for LP-solving. Let $\{x'_S \mid S \in \mathcal{R}'\}$ be the resulting solution. From the Strong Duality Theorem, we get that:

$$\sum_{S\in\mathcal{R}'} m(S)\cdot x'_S = \mathrm{OPT}(\mathcal{R}').$$

From Claim 4.8, with high probability:

$$\mathrm{OPT}(\mathcal{R}') \geq \Omega\left(\frac{\mathrm{OPT}_{\text{LP-D}}}{\beta(N)}\right) = \Omega\left(\frac{\mathrm{OPT}_{\text{LP-P}}}{\beta(N)}\right).$$

Altogether, we get that with high probability, $\sum_{S\in\mathcal{R}'} m(S)\cdot x'_S \geq \Omega\left(\frac{\mathrm{OPT}_{\text{LP-P}}}{\beta(N)}\right)$. We can extend the solution $\{x'_S \mid S\in\mathcal{R}'\}$ to (LP-P2) to obtain a feasible solution $\{x'_S \mid S\in\mathcal{R}\}$ to (LP-P) by setting the value x'_S for all sets $S\in\mathcal{R}\setminus\mathcal{R}'$ to 0. It is immediate to verify that the resulting solution to (LP-P) is feasible, and its value remains unchanged. Therefore, we have obtained a randomized algorithm, with running time bounded by $O(\mathrm{poly}(N))$, that with high probability computes a solution to (LP-P), whose value is at least $\Omega\left(\frac{\mathrm{OPT}_{\text{LP-P}}}{\beta(N)}\right)$; here, $\beta(N) = O(\alpha(N^2)\cdot\log^2 N)$.

We are now ready to complete the reduction from Dense k-Coloring to Densest k-Subgraph from Theorem 4.1. Let $\alpha:\mathbb{Z}^+\to\mathbb{Z}^+$ be an increasing function, such that $\alpha(n)\leq o(n)$, and 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.

Consider now the input instance $\mathrm{DkC}(G,k)$ of the Dense k-Coloring problem with $N=|V(G)|$. We can assume w.l.o.g. that N is greater than a large enough constant, as otherwise we can solve the problem exactly via exhaustive search. Since $\alpha(n)\leq o(n)$, we can also assume that $\beta(N)\leq N^3$. We use the randomized algorithm described above, that, in time $O(\mathrm{poly}(N))$, with high probability computes a $\beta(N)$-approximate solution to (LP-P). Recall that the number of variables of (LP-P) with non-zero LP-value is bounded by $O(\mathrm{poly}(N))$. Next, we apply the algorithm from Claim 4.2 in order to round the resulting LP solution. The algorithm with high probability returns an integral solution $(S_1,\ldots,S_{N/k})$ to instance $\mathrm{DkC}(G,k)$, such that $\sum_{i=1}^{N/k}|E_G(S_i)| \geq \Omega\left(\frac{\mathrm{OPT}_{\mathrm{DkC}}(G,k)}{\beta(N)\cdot\log^3 N}\right)$. Since $\beta(N)=O(\alpha(N^2)\cdot\log^2 N)$, with high probability we obtain an $O(\alpha(N^2)\cdot\mathrm{poly}\log N)$-approximate solution to the input instance of Dense k-Coloring. In order to complete the reduction from Dense k-Coloring to Densest k-Subgraph from Theorem 4.1, it is now enough to prove Claim 4.8, which we do next. The proof uses standard techniques and is only included for completeness.

Proof of Claim 4.8. Let $m=|E(G)|$. Notice that we can assume w.l.o.g. that in an optimal solution to (LP-D), for every vertex $v\in V(G)$, $y_v\leq m$ holds, and $z\leq m$. Indeed, if this is not the case, then we can modify the solution by setting $y_v=\min\{y_v,m\}$ for every vertex $v\in V(G)$, and $z=\min\{z,m\}$. It is easy to verify that this remains a feasible solution, and its value does not grow. For convenience, for every subset $\mathcal{R}'\subseteq\mathcal{R}$ of vertex subsets, we define the following LP:

$$
\begin{aligned}
&(P'(\mathcal{R}')) \\
&\min \quad \tfrac{N}{k}\cdot z + \textstyle\sum_{v\in V(G)} y_v \\
&\text{s.t.} \\
&\qquad z + \textstyle\sum_{v\in S} y_v \geq m(S) \quad \forall S\in\mathcal{R}' \\
&\qquad\quad 0\leq z\leq m \\
&\qquad\quad 0\leq y_v\leq m \qquad \forall v\in V(G)
\end{aligned}
$$

We denote by $\mathrm{OPT}'(\mathcal{R}')$ the value of the optimal solution of the above LP. From the above discussion, it is enough to provide a randomized algorithm with running time $O(\mathrm{poly}(N))$, that computes a collection $\mathcal{R}'\subseteq\mathcal{R}$ of subsets of vertices, such that, with high probability, $\mathrm{OPT}'(\mathcal{R}')\geq \Omega\left(\mathrm{OPT}'(\mathcal{R})/\beta(N)\right)$, where $\beta(N)=O(\alpha(N^2)\cdot\log^2 N)$. This is since $\mathrm{OPT}(\mathcal{R}')=\mathrm{OPT}'(\mathcal{R}')$, and $\mathrm{OPT}'(\mathcal{R})=\mathrm{OPT}(\mathcal{R})$ holds.

Clearly, $\mathrm{OPT}'(\mathcal{R})\leq 2Nm$ must hold. Let M be the smallest integral power of 2 that is greater than $2Nm$. For all $0<C\leq\log M$ and $\mathcal{R}'\subseteq\mathcal{R}$, we consider a feasibility LP, that is obtained from $P'(\mathcal{R}')$, by adding the constraint that $\frac{N}{k}\cdot z+\sum_{v\in V(G)}y_v\leq 2^C$:

$$
\begin{aligned}
&(F(\mathcal{R}',C)) \\
&\qquad \tfrac{N}{k}\cdot z + \textstyle\sum_{v\in V(G)} y_v \leq 2^C \\
&\qquad z + \textstyle\sum_{v\in S} y_v \geq m(S) \qquad \forall S\in\mathcal{R}' \\
&\qquad\quad 0\leq z\leq m \\
&\qquad\quad 0\leq y_v\leq m \qquad \forall v\in V(G)
\end{aligned}
$$

The key to the proof of Claim 4.8 is the following observation.

Observation 4.9. *Assume that there is an efficient $\alpha(n)$-approximation algorithm for* Densest k-Subgraph*, where α is an increasing function of n, and n is the number of vertices in the input graph. Then there is a randomized algorithm with running time $O(\mathrm{poly}(N))$, that, given a value $0\leq C\leq\log M$, either:*

- *computes a collection $\mathcal{R}'(C)\subseteq\mathcal{R}$ of at most $O(\mathrm{poly}(N))$ subsets of vertices, such that the linear program $(F(\mathcal{R}',C))$ is infeasible; or*
- *computes values $0\leq z'\leq m$ and $0\leq y'_v\leq m$ for all $v\in V(G)$, such that $\frac{N}{k}\cdot z+\sum_{v\in V(G)}y_v\leq 2^C$, and, with high probability, for every vertex set $S\in\mathcal{R}$, $z'+\sum_{v\in S}y'_v\geq m(S)/\beta(N)$.*

We provide the proof of Observation 4.9 below, after we complete the proof of Claim 4.8 using it. We apply the algorithm from Observation 4.9 to every value $0\leq C\leq\log M$. We say that the application of the algorithm for value C is *successful* if the algorithm returns values $0\leq z'\leq m$ and $0\leq y'_v\leq m$ for all $v\in V(G)$; otherwise we say that it is unsuccessful. We say that the algorithm *errs* if it is successful, and yet there is a set $S\in\mathcal{R}$ of vertices for which $z'+\sum_{v\in S}y'_v<m(S)/\beta(N)$.

Let C^* be the smallest value of C, such that the algorithm from Observation 4.9, when applied to C, was successful. Let z' and $\{y'_v \mid v \in V(G)\}$ be the values of the LP-variables returned by the algorithm. We let $\hat{\mathcal{E}}$ be the bad event that the algorithm from Observation 4.9, when applied to value C^*, errs. The probability of $\hat{\mathcal{E}}$ is at most $1/\mathrm{poly}(N)$. Consider the following solution to $(P'(\mathcal{R}))$: we set $z^* = \min\{z' \cdot \beta(N), m\}$, and for all $v \in V(G)$, we set $y^*_v = \min\{y'_v \cdot \beta(N), m\}$. It is immediate to verify that, if Event $\hat{\mathcal{E}}$ did not happen, then we obtain a feasible solution to $(P'(\mathcal{R}))$, whose value is at most $2^{C^*} \cdot \beta(N)$. Notice that, from the choice of the value C^*, LP $(F(\mathcal{R}'(C^*-1), C^*-1))$ does not have a feasible solution. Since the constraints in $(F(\mathcal{R}'(C^*-1), C^*-1)$ are a subset of the constraints in $(F(\mathcal{R}, C^*-1))$, it follows that $(F(\mathcal{R}, C^*-1))$ does not have a feasible solution, and so $\mathrm{OPT}'(\mathcal{R}) \geq 2^{C^*-1}$. Therefore, if Event $\hat{\mathcal{E}}$ did not happen, we obtain a feasible solution $(z^*, \{y^*_v\}_{v \in V(G)})$ to $(P'(\mathcal{R}))$, whose value is at most $2\beta(N) \cdot \mathrm{OPT}'(\mathcal{R})$.

The final collection of subsets of vertices that our algorithm returns is $\mathcal{R}' = \bigcup_{C=0}^{C^*-1} \mathcal{R}'(C)$. Clearly, $|\mathcal{R}'| \leq \mathrm{poly}(N)$. It is also immediate to verify that $\mathrm{OPT}(\mathcal{R}') \geq 2^{C^*-1}$. Indeed, for all values $0 \leq C < C^*$, linear program $(F(\mathcal{R}'(C), C))$ is infeasible, and, since $\mathcal{R}'(C) \subseteq \mathcal{R}'$, linear program $(F(\mathcal{R}', C))$ is also infeasible. Since every constraint of $P'(\mathcal{R}')$ is also a constraint of $LP(\mathcal{R})$, we get that, if Event $\hat{\mathcal{E}}$ did not happen, then $(z^*, \{y^*_v\}_{v \in V(G)})$ is a feasible solution to $P'(\mathcal{R}')$, whose value is at most $\beta(N) \cdot 2^{C^*}$. To summarize, if Event $\hat{\mathcal{E}}$ did not happen, then:

$$2^{C^*-1} \leq \mathrm{OPT}'(\mathcal{R}') \leq 2^{C^*} \cdot \beta(N)$$

and

$$2^{C^*-1} \leq \mathrm{OPT}'(\mathcal{R}) \leq 2^{C^*} \cdot \beta(N)$$

hold.

Therefore, if Event $\hat{\mathcal{E}}$ did not happen, then $\mathrm{OPT}'(\mathcal{R}') \geq 2^{C^*-1} \geq \mathrm{OPT}'(\mathcal{R})/(2\beta(N))$. It now remains to prove Observation 4.9. The proof is standard; we only provide its sketch below.

Proof of Observation 4.9. We fix a value $0 \leq C \leq \log M$, and consider the corresponding Linear Program $(F(\mathcal{R}, C))$. The idea of the proof is simple: we employ the Ellipsoids algorithm, together with the separation oracle from Lemma 4.4 (after we reduce its error probability by repeating the algorithm a number of times). We then let $\mathcal{R}'(C)$ be the collection of all vertex subsets $S \in \mathcal{R}$, such that the separation oracle returns the constraint associated with S over the course of the algorithm.

We now provide more details. Recall first the Ellipsoids algorithm for solving a feasibility Linear Program (F) on N' variables. The algorithm proceeds in iterations. The input to the ith iteration is an N'-dimensional Ellipsoid E_i, that contains the feasible region of (F). Let x_i denote the center point of the ellipsoid. If the algorithm is given a constraint A^j of the Linear Program (F) that is violated by point x_i, then it produces a new ellipsoid E_{i+1}, that contains the feasible region of (F), whose volume is at most $(1 - 1/\mathrm{poly}(N'))$ times the volume of E_i. The running time of a single iteration is $O(\mathrm{poly}(N'))$.

Typically, we assume that there is an initial ellipsoid E_1, whose volume is at most $2^{\mathrm{poly}(N')}$, that contains the feasible region of (F), which needs to be supplied to the Ellipsoids algorithm. We can also typically assume that, if (F) has a feasible solution, then the volume of the feasible region of (F) is at least $L = 2^{-\mathrm{poly}(N')}$ (if this is not the case, the feasible region can be slightly inflated artificially

by adding a small amount of slack to the constraints; in our case, since we are only solving the LP approximately, this is immaterial). If the above two conditions hold, the algorithm can proceed for at most $\text{poly}(N')$ iterations, before the volume of the current ellipsoid becomes smaller than L, and the algorithm then correctly declares that (F) does not have a feasible solution. In every iteration, the constraint violated by the center x_i of the current ellipsoid E_i is supplied by a separation oracle. If the separation oracle declares that x_i is (approximately) feasible solution, then the algorithm halts.

We now turn to consider the linear program $(F(\mathcal{R}, C))$. Since the LP constraints require that the values of all LP-variables are between 0 and m, it is easy to verify that the feasible region of the LP is contained in the $(N+1)$-dimensional sphere E_1, whose radius is bounded by $\text{poly}(m)$, and volume is at most $2^{\text{poly}(m)} \leq 2^{\text{poly}(N)}$. We initially set $\mathcal{R}'(C) = \emptyset$. We apply the Ellipsoids algorithm to this LP, with the initial ellipsoid E_1.

We now consider the ith iteration of the algorithm, whose input is an ellipsoid E_i, together with its center point $(z^i, \{y_v^i\}_{v \in V(G)})$. We manually check the constraints $\frac{N}{k} \cdot z^i + \sum_{v \in V(G)} y_v^i \leq 2^C$; $0 \leq z^i \leq m$; and $0 \leq y_v^i \leq m$ for all $v \in V(G)$. If any of these constraints does not hold, then we return it as a violated constraint. Assume now that all these constraints hold. We apply the algorithm from Lemma 4.4 to the current values $(z^i, \{y_v^i\}_{v \in V(G)})$; we do so N times. If, in each of these iterations, the algorithm returns "accept", then we terminate our algorithm, and return the current solution $(z^i, \{y_v^i\}_{v \in V(G)})$. Observe that we are guaranteed that $0 \leq z^i \leq m$; $0 \leq y_v^i \leq m$ for all $v \in V(G)$; and $\frac{N}{k} \cdot z^i + \sum_{v \in V(G)} y_v^i \leq 2^C$. Moreover, unless the algorithm from Lemma 4.4 erred in each of its N applications, we are guaranteed that, for every vertex set $S \in \mathcal{R}$, $z^i + \sum_{v \in S} y_v^i \geq m(S)/\beta(N)$ holds. The probability that the algorithm from Lemma 4.4 errs is at most $2/3$, so with high probability, we are guaranteed that for all $S \in \mathcal{R}$, $z^i + \sum_{v \in S} y_v^i \geq m(S)/\beta(N)$.

Assume now that in some application of the algorithm from Lemma 4.4 to the current values $(z^i, \{y_v^i\}_{v \in V(G)})$ we obtain a violated constraint of (LP-D). That is, we obtain a set $S \in \mathcal{R}$ of vertices, for which $z^i + \sum_{v \in S} y_v^i < m(S)$ holds. In this case, we add S to set $\mathcal{R}'(S)$, and we use this constraint as a violated constraint for the Ellipsoids algorithm.

If the above algorithm never terminates with an approximately feasible solution $(z^i, \{y_v^i\}_{v \in V(G)})$, then we are guaranteed that after at most $\text{poly}(N)$ iterations, the algorithm correctly certifies that $(F(\mathcal{R}, C))$ does not have a feasible solution. We then return the current collection $\mathcal{R}(C)$ of vertex subsets. For convenience, we denote by $A^1, A^1, \ldots, A^r$ the sequence of violated constraints that were fed to the Ellipsoids algorithm. Each of the constraints A^j either corresponds to a set $S \in \mathcal{R}(C)$, or it is one of the constraints $0 \leq z^i \leq m$; $0 \leq y_v^i \leq m$ for all $v \in V(G)$; and $\frac{N}{k} \cdot z^i + \sum_{v \in V(G)} y_v^i \leq 2^C$. In other words, each such constraint A^j is also a constraint of the LP $(F(\mathcal{R}(C), C))$.

It now remains to prove that in the latter case, $(F(\mathcal{R}(C), C))$ does not have a feasible solution. In order to do so, consider applying the Ellipsoids algorithm to this linear program. We start with the same initial ellipsoid E_1 as before. Since the Ellipsoid algorithm is deterministic, its behavior is entirely determined by the initial ellipsoid E_1 and the sequence of the violated constraints that it receives. We will use exactly the same sequence $A^1, A^1, \ldots, A^r$ of violated constraints in this execution of Ellipsoids algorithm. This ensures that for all i, the ellipsoid E_i that is used as the input to the ith iteration is identical to the ellipsoid that was used as input to iteration i when solving $(F(\mathcal{R}, C))$, which in turn ensures that constraint A^i is a violating constraint for the center of ellipsoid E_i. Therefore, this execution of Ellipsoids algorithm is identical to the execution of the

same algorithm when applied to LP $(F(\mathcal{R}, C))$, and it will end up with a final ellipsoid E_r, whose volume is small enough to correctly establish that $(F(\mathcal{R}(C), C))$ does not have a feasible solution. □ □

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

In this section we prove the following theorem.

Theorem 5.1. *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 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.

We obtain the following immediate corollary of Theorem 5.1.

Corollary 5.2. *Assume that Conjecture 2 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.

Proof: We prove the corollary for (r,h)-Graph Partitioning; the proof for Dense k-Coloring is similar. Assume that Conjecture 2 holds and that NP $\not\subseteq$ DTIME$(n^{O(\log n)})$. Then, from Theorem 3.2, for some constant $0 < \varepsilon < 1$, there is no randomized factor-$2^{(\log n)^{\varepsilon}}$-approximation algorithm for Densest k-Subgraph with running time $n^{O(\log n)}$, where n is the number of vertices in the input graph.

We let $\varepsilon' = \varepsilon/c$, where c is a sufficiently large constant. We now prove that there is no efficient $2^{(\log n)^{\varepsilon'}}$-approximation algorithm for (r,h)-Graph Partitioning. Indeed, assume for contradiction that there is an efficient $2^{(\log n)^{\varepsilon'}}$-approximation algorithm $\mathcal{A}$ for (r,h)-Graph Partitioning. From Theorem 5.1, there is a randomized algorithm for the Densest k-Subgraph problem, that, given an instance DkS(G, k) of the problem with $|V(G)| = N$, in time $N^{O(\log N)}$, computes a $c'(\alpha(N^{O(\log N)}))^3 \cdot \log^2 N$-approximate solution, where $\alpha(x) = 2^{(\log x)^{\varepsilon'}}$ and c' is a constant independent of N. Note that:

$$\alpha(N^{O(\log N)}) = 2^{(\log N)^{O(\varepsilon')}} = 2^{(\log N)^{O(\varepsilon/c)}}.$$

Since we can let c be a sufficiently large constant, we can ensure that $c'(\alpha(N^{O(\log N)}))^3 \cdot \log^2 N < 2^{(\log n)^{\varepsilon}}$.

Therefore, we obtain a randomized factor-$2^{(\log n)^{\varepsilon}}$-approximation algorithm for Densest k-Subgraph, with running time $n^{O(\log n)}$, a contradiction. □

The remainder of this section is dedicated to the proof of Theorem 5.1. In order to obtain both reductions, we start with an instance $\mathrm{DkS}(G,k)$ of the Densest k-Subgraph problem, and construct another auxiliary graph H. This graph is then used in order to define the corresponding instances of Dense k-Coloring and (r,h)-Graph Partitioning, respectively. We start by defining graph H and analyzing its properties in Section 5.1. We then complete the reduction from Densest k-Subgraph to Dense k-Coloring in Section 5.2, and the reduction from Densest k-Subgraph to (r,h)-Graph Partitioning in Section 5.3. Throughout this section, for an integer A, we denote $[A] = \{0, 1, \ldots, A-1\}$. We also assume that the parameter k in the input instance of the Densest k-Subgraph problem is greater than a large enough constant, since otherwise the problem can be solved in time $\mathrm{poly}(N)$ via exhaustive search.

5.1 Auxiliary Graph H

Let $\mathrm{DkS}(G,k)$ be an instance of the Densest k-Subgraph problem. Denote $V(G) = \{v_0, v_1, \ldots, v_{N-1}\}$. We now provide a randomized algorithm to construct an auxiliary graph H corresponding to this instance. The construction is somewhat similar to and inspired by the construction used in Section 2 of [KLS00].

Let $q = \lceil \log N \rceil$. We start by computing a prime number M, such that $N^{5q} \leq M \leq 2 \cdot N^{5q}$. From the Bertrand-Chebyshev theorem [Ber45, Čeb50], such a prime number must exist, and it can be computed in time $N^{O(\log N)}$ by checking every integer between N^{5q} and $2 \cdot N^{5q}$. We then construct a random mapping $f : [N] \to [M]$ as follows. For every integer $1 \leq i \leq N$, we let $f(i)$ be an integer chosen independently and uniformly at random (with replacement) from $[M]$.

We are now ready to define the graph H. The set of vertices of H is $V(H) = \{u_0, \ldots, u_{M-1}\}$. For every edge $e = (v_i, v_j) \in E(G)$, we construct a collection $J(e)$ of M edges in H: $J(e) = \{\left(u_{f(i)+t}, u_{f(j)+t}\right) \mid 0 \leq t \leq M-1\}$, where the addition in the subscript is modulo M (we use this convention throughout the remainder of this section). We say that edge e is the *origin* of every edge in set $J(e)$. We then set $E(H) = \bigcup_{e \in E(G)} J(e)$. We note that we do not allow parallel edges in H, so it is possible for an edge in H to have several origin edges in G. This completes the definition of the graph H. We now analyze its properties.

Good Event $\mathcal{E}^g$. We say that a good event $\mathcal{E}^g$ happens if there is a collection $\{H_1, \ldots, H_r\}$ of $r = \left\lfloor \frac{M}{k \log k} \right\rfloor$ disjoint subgraphs of H, such that the following hold:

- for all $1 \leq j \leq r$, $|V(H_j)| = k$;
- for all $1 \leq j \leq r$, $|E(H_j)| \leq \mathrm{OPT}_{\mathrm{DkS}}(G,k)$; and
- $\sum_{1 \leq j \leq r} |E(H_j)| \geq 0.1 \cdot \left\lfloor \frac{M}{k \log k} \right\rfloor \cdot \mathrm{OPT}_{\mathrm{DkS}}(G,k)$.

We start by showing that good event $\mathcal{E}^g$ happens with a sufficiently high probability.

Claim 5.3. $\mathbf{Pr}\left[\mathcal{E}^g\right] \geq 0.8$.

Proof: Let S be the optimal solution to instance $\mathrm{DkS}(G,k)$ of the Densest k-Subgraph problem, and let T be a subset of vertices of H, defined as $T = \{u_{f(i)} \mid v_i \in S\}$. Let I be a collection of $r = \lfloor M/(k \log k) \rfloor$ integers from $[M]$, obtained by selecting each integer independently uniformly at random (with replacement) from $[M]$. For every index $j \in I$, we define a set T'_j of vertices of H as follows: $T'_j = \{u_{f(i)+j} \mid v_i \in S\}$. Finally, for every index $j \in I$, we define another set T_j of vertices of H, by starting with the set T'_j of vertices, and then removing from it every vertex that lies in set $\bigcup_{i \in I \setminus \{j\}} T'_i$. Clearly, all resulting vertex sets in $\{T_j\}_{j \in I}$ are mutually disjoint, and each such set contains at most k vertices.

For every index $j \in I$, we denote by $\hat{E}(T_j)$ the set of edges $e \in E_H(T_j)$, such that an origin of e in G lies in $E_G(S)$. Equivalently: $\hat{E}(T_j) = \{(u_{f(i)+j}, u_{f(i')+j}) \mid (v_i, v_{i'}) \in E_G(S)\}$. Clearly, $|\hat{E}(T_j)| \leq \mathrm{OPT}_{\mathrm{DkS}}(G,k)$ for all $j \in I$, and so $\sum_{j \in I} |\hat{E}(T_j)| \leq r \cdot \mathrm{OPT}_{\mathrm{DkS}}(G,k) = \left\lfloor \frac{M}{k \log k} \right\rfloor \cdot \mathrm{OPT}_{\mathrm{DkS}}(G,k)$.

We prove the following observation.

Observation 5.4.

$$\mathbf{E}\left[\sum_{j \in I} |\hat{E}(T_j)|\right] \geq 0.9 \cdot \left\lfloor \frac{M}{k \log k} \right\rfloor \cdot \mathrm{OPT}_{\mathrm{DkS}}(G,k).$$

Assume first that the observation holds. For simplicity of notation, denote $B = \left\lfloor \frac{M}{k \log k} \right\rfloor \cdot \mathrm{OPT}_{\mathrm{DkS}}(G,k)$. For all $j \in I$, we define a subgraph H_j of H, whose vertex set is T_j, and edge set is $\hat{E}(T_j)$. Clearly, the graphs in $\{H_j \mid j \in I\}$ are disjoint, and, for all $j \in I$, $|V(H_j)| = |T| \leq k$ holds. Additionally, from the above discussion, for all $j \in I$, $|E(H_j)| \leq \mathrm{OPT}_{\mathrm{DkS}}(G,k)$. If, additionally, $\sum_{j \in I} |\hat{E}(T_j)| \geq 0.1B$ holds, then Event $\mathcal{E}^g$ happens. As observed above, $\sum_{j \in I} |\hat{E}(T_j)| \leq B$ must hold. Let p denote the probability that $\sum_{j \in I} |\hat{E}(T_j)| \geq 0.1B$. Then:

$$\mathbf{E}\left[\sum_{j \in I} |\hat{E}(T_j)|\right] \leq (1-p) \cdot 0.1 \cdot B + p \cdot B = 0.1B + 0.9Bp.$$

Since, from Observation 5.4, $\mathbf{E}\left[\sum_{j \in I} |\hat{E}(T_j)|\right] \geq 0.9B$, we conclude that $p \geq 0.8$, and so $\mathbf{Pr}\left[\mathcal{E}^g\right] \geq 0.8$, as required. In order to complete the proof of Claim 5.3, it is now enough to prove Observation 5.4, which we do next.

Proof of Observation 5.4. We associate a collection $\{X_1, \ldots, X_r\}$ of random variables with the set I of indices. In order to do so, we view set I as being constructed as follows. For each $1 \leq i \leq r$, sample a value X_i uniformly at random from $[M]$, and then let $I = \{X_1, \ldots, X_r\}$ be the collection of these sampled values. Consider now any index $1 \leq i \leq r$, the corresponding set T'_{X_i} of vertices of H, and any edge $e = (v_a, v_b) \in E_G(S)$. Edge $e' = (u_{f(a)+X_i}, u_{f(b)+X_i})$ of H corresponding to e belongs to set $\hat{E}(T_{X_i})$ if and only if neither of the vertices $u_{f(a)+X_i}$, $u_{f(b)+X_i}$ lies in $\bigcup_{j \in I \setminus \{i\}} T'_{X_j}$. Consider now an index $j \in I \setminus \{i\}$. The probability that a fixed vertex $u \in V(H)$ lies in T'_{X_j} is at most k/M

(since for every vertex $v_z \in S$, there is a single index s with $u_{f(z)+s} = u$). From the union bound, the probability that a fixed vertex $u \in V(H)$ lies in $\bigcup_{j \in I \setminus \{i\}} T'_{X_j}$ is at most $\frac{r \cdot k}{M} \leq \frac{1}{\log k}$. In particular, the probability that any of the endpoints of edge $e' = (u_{f(a)+X_i}, u_{f(b)+X_i})$ lies in $\bigcup_{j \in I \setminus \{i\}} T'_{X_j}$ is at most $\frac{2}{\log k}$. Therefore, $\mathbf{E}\left[|\hat{E}(T_j)|\right] \geq |E_G(S)| \cdot \left(1 - \frac{2}{\log k}\right) \geq 0.9 \cdot |E_G(S)|$. Altogether, from the linearity of expectation, $\mathbf{E}\left[\sum_{1 \leq i \leq r} |\hat{E}(T_{X_i})|\right] \geq 0.9 \cdot |E_G(S)| \cdot r = 0.9 \cdot \left\lfloor \frac{M}{k \log k} \right\rfloor \cdot \mathrm{OPT}_{\mathrm{DkS}}(G, k)$. □□

Ensemble and Bad Event $\mathcal{E}^b$. Next, we define the notion of an ensemble. Recall that $q = \lceil \log N \rceil$. An ensemble $\mathcal{B}$ consists of a collection $I \subseteq [N]$ of at most $2q$ indices, and, for every index $i \in I$, an integer $-q \leq x_i \leq q$ with $x_i \neq 0$. We denote the ensemble by $\mathcal{B} = \{I, \{x_i\}_{i \in I}\}$. We say that ensemble $\mathcal{B} = \{I, \{x_i\}_{i \in I}\}$ is *bad* if $\sum_{i \in I} x_i \cdot f(i) \equiv 0 \pmod{M}$. We let $\mathcal{E}^b$ be the bad event that there exists a bad ensemble. We start by showing that the probability of Event $\mathcal{E}^b$ happening is low. Later, we show that, if Event $\mathcal{E}^b$ does not happen, then graph H has some useful properties.

Observation 5.5. $\mathbf{Pr}\left[\mathcal{E}^b\right] \leq \frac{1}{N^q}$.

Proof: Consider any fixed ensemble $\mathcal{B} = \{I, \{x_i\}_{i \in I}\}$. Let $i^* \in I$ be any fixed index. Consider now the following two-step process: in the first step, we select the values $f(i)$ for all indices $i \in I \setminus \{i^*\}$ independently uniformly at random from $[M]$. We then denote $S = \sum_{i \in I \setminus \{i^*\}} x_i \cdot f(i)$. In the second step, we select the value $a = f(i^*)$ at random from $[M]$. Ensemble $\mathcal{B}$ is bad if and only if $a \cdot x_{i^*} + S = 0 \mod M$. Since M is a prime number, there is exactly one value $a' \in [M]$ with $a' \cdot x_{i^*} + S = 0 \mod M$ (indeed, if two such values a', a'' exist, then $a' \cdot x_{i^*} = a'' \cdot x_{i^*} \mod M$, implying that $a' = a''$ must hold). The probability to choose $f(i^*) = a'$ is then $1/M$, and so the probability that a fixed ensemble $\mathcal{B}$ is bad is $1/M$.

Notice that the total number of ensembles is bounded by $\left(\sum_{t=1}^{2q} \binom{N}{t}\right) \cdot (2q)^{2q} \leq (2q) \cdot \binom{N}{2q} \cdot (2q)^{2q} \leq N^{4q}$. Using the Union Bound, $\mathbf{Pr}\left[\mathcal{E}^b\right] \leq \frac{N^{4q}}{M} \leq \frac{1}{N^q}$, since $M \geq N^{5q}$. □

Next, we show that, if Event $\mathcal{E}^b$ does not happen, then every edge $e \in E(H)$ has a unique origin edge in G.

Observation 5.6. *Assume that Event $\mathcal{E}^b$ did not happen. Let e be any edge of H. Then there is a unique edge $e' \in E(G)$, such that e' is an origin edge of e.*

Proof: Denote $e = (u_j, u_{j'})$, and assume for contradiction that there are two distinct edges $e_1, e_2 \in E(G)$ that both serve as origin edges of e. Denote $e_1 = (v_{i_1}, v_{i'_1})$ and $e_2 = (v_{i_2}, v_{i'_2})$. From the construction of H, since edge e_1 is an origin edge of e, there exists an integer t_1, such that:

$$j \equiv f(i_1) + t_1 \pmod{M}, \quad \text{and} \quad j' \equiv f(i'_1) + t_1 \pmod{M}. \tag{5}$$

Similarly, since edge e_2 is an origin edge of e, there exists an integer t_2, such that:

$$j \equiv f(i_2) + t_2 \pmod{M}, \quad \text{and} \quad j' \equiv f(i'_2) + t_2 \pmod{M}. \tag{6}$$

By adding the first equation of (5) to the second equation of (6), we get that $f(i_1) + f(i'_2) \equiv j + j' - t_1 - t_2 \pmod{M}$. Similarly, by adding the second equation of (5) to the first equation of (6), we get that $f(i'_1) + f(i_2) \equiv j + j' - t_1 - t_2 \pmod{M}$. In other words, we get that:

$$f(i_1) + f(i'_2) \equiv f(i'_1) + f(i_2) \pmod{M}. \tag{7}$$

Since e_1 is an edge of G, $i_1 \neq i'_1$ must hold, and similarly, since e_2 is an edge of G, $i_2 \neq i'_2$ must hold. Moreover, since $e_1 \neq e_2$, either $i_1 \neq i_2$, or $i'_1 \neq i'_2$ must hold. Combining this with Equation (7), we conclude that both $i_1 \neq i_2$ and $i'_1 \neq i'_2$ must hold.

We now consider four cases. The first case is when all indices in $\{i_1, i'_1, i_2, i'_2\}$ are distinct. In this case, we consider the ensemble $\mathcal{B} = \{I, \{x_i\}_{i \in I}\}$, where $I = \{i_1, i'_1, i_2, i'_2\}$, with $x_{i_1} = x_{i'_2} = 1$ and $x_{i_2} = x_{i'_1} = -1$. From Equation 7, we get that $\sum_{i \in I} x_i \cdot f(i) \equiv 0 \pmod{M}$, so ensemble $\mathcal{B}$ is bad, contradicting the fact that Event $\mathcal{E}^b$ did not happen.

The second case is when $i_1 = i'_2$ but $i'_1 \neq i_2$. Then we construct an ensemble $\mathcal{B} = \{I, \{x_i\}_{i \in I}\}$, where $I = \{i_1, i'_1, i_2\}$, with $x_{i_1} = 2$ and $x_{i_2} = x_{i'_1} = -1$. As before, from Equation 7, we get that $\sum_{i \in I} x_i \cdot f(i) \equiv 0 \pmod{M}$, so ensemble $\mathcal{B}$ is bad, contradicting the fact that Event $\mathcal{E}^b$ did not happen.

The third case is when $i'_1 = i_2$ but $i_1 \neq i'_2$. We consider the ensemble $\mathcal{B} = \{I, \{x_i\}_{i \in I}\}$, where $I = \{i_1, i'_1, i'_2\}$, with $x_{i_1} = x_{i'_2} = 1$ and $x_{i_2} = -2$. From Equation 7, we get that $\sum_{i \in I} x_i \cdot f(i) \equiv 0 \pmod{M}$, so ensemble $\mathcal{B}$ is bad, contradicting the fact that Event $\mathcal{E}^b$ did not happen.

From the above discussion, the only remaining case is when both $i'_1 = i_2$ and $i_1 = i'_2$ hold. But in this case, $e_1 = e_2$, contradicting our assumption that these two edges are distinct. □

Assume that bad event $\mathcal{E}^b$ did not happen. For every edge $e \in E(H)$, we denote by $R(e)$ the unique edge of G that serves as the origin edge of e. For a subgraph $H' \subseteq H$, we let $R(H')$ be the subgraph of G induced by the set $\{R(e) \mid e \in E(H')\}$ of edges; we refer to $R(H')$ as the *origin graph of* H'. In other words, the set of edges of graph $R(H')$ is $\{R(e) \mid e \in E(H')\}$, and the set of its vertices contains every vertex of G that serves as an endpoint to any of these edges.

In the next observation we show that, if bad Event $\mathcal{E}^b$ did not happen, then for every cycle $C \subseteq H$ containing at most q edges, every vertex in the corresponding origin-graph $R(C)$ has an even degree.

Observation 5.7. *Assume that Event $\mathcal{E}^b$ did not happen. Let $C \subseteq H$ be any simple cycle containing at most q edges, and let $G' = R(C)$ be the origin graph of C. Then every vertex of G' has an even degree in G'.*

Proof: Throughout the proof, we assume that Event $\mathcal{E}^b$ did not happen, and we fix a simple cycle $C = (u_{j_1}, \ldots, u_{j_z})$ in H, with $z \leq q$. For all $1 \leq i \leq z$, we denote $e_i = (u_{j_i}, u_{j_{i+1}})$, and we denote the origin-edge of e_i by $e'_i = R(e_i) = (v_{a_i}, v_{b_i})$. From the definition of graph H, there must be an integer $t_i \in [M]$, with $j_i \equiv f(a_i) + t_i \pmod{M}$ and $j_{i+1} \equiv f(b_i) + t_i \pmod{M}$. Therefore, for all $1 \leq i \leq z$:

$$f(a_i) - f(b_i) \equiv j_i - j_{i+1} \pmod{M}.$$

Summing up the above equality over all $i = 1, \ldots z$, we get that

$$\sum_{1 \leq i \leq z} f(a_i) - \sum_{1 \leq i \leq z} f(b_i) \equiv 0 \pmod{M}. \tag{8}$$

Let A be the set of indices lying in $\{a_1, \ldots, a_z\}$ (if an index appears several times in $\{a_1, \ldots, a_z\}$, we only include it once in A). For every index $a^* \in A$, let x'_{a^*} be the number of integers $i \in \{1, \ldots, z\}$ with $a_i = a^*$. Similarly, we let B be the set of indices lying in $\{b_1, \ldots, b_z\}$, and for every index $b^* \in B$, we let x''_{b^*} be the number of integers $i \in \{1, \ldots, z\}$ with $b_i = a$. We claim that $A = B$ must hold, and, for every index $a \in A$, $x'_a = x''_a$ must hold. Indeed, assume otherwise. We then construct an ensemble $\mathcal{B} = \{I, \{x_i\}_{i \in I}\}$ as follows. Set I includes every index $i \in (A \setminus B) \cup (B \setminus A)$; for each such index i, we set $x_i = x'_i$ if $i \in A \setminus B$ and $x_i = -x''_i$ otherwise. Additionally, for every index $i \in A \cap B$ with $x'_i \neq x''_i$, we include index i in I, with $x_i = x'_i - x''_i$. From our assumptions, $I \neq \emptyset$, $|I| \leq 2q$, and for all $i \in I$, $-q \leq x_i \leq q$, with $x_i \neq 0$. Therefore, $\mathcal{B} = \{I, \{x_i\}_{i \in I}\}$ is a valid ensemble. But then, from Equation 8, $\sum_{i \in I} x_i \cdot f(i) \equiv 0 \pmod{M}$. In other words, ensemble $\mathcal{B}$ is bad, contradicting the assumption that bad event $\mathcal{E}^b$ did not happen.

We conclude that $A = B$ must hold, and, for every index $a \in A$, $x'_a = x''_a$ must hold. Therefore, for every vertex $v \in V(G)$, the number of times that v lies in $\{v_{a_1}, \ldots, v_{a_z}\}$ is equal to the number of times that v lies in $\{v_{b_1}, \ldots, v_{b_z}\}$. Therefore, the number of edges of $\{R(e_i) \mid 1 \leq i \leq z\}$ that are incident to v is even. □

Lastly, we need the following claim.

Claim 5.8. *Let H' be any subgraph of H with $|V(H')| \leq N$. If Event $\mathcal{E}^b$ did not happen, then the origin graph $R(H')$ of H' contains at most $c^* \cdot |V(H')|$ vertices, where c^* is a constant independent of N.*

Proof: Recall that the *girth* of an unweighted graph G^* is the length of the shortest cycle in G^*. For an integer $t \geq 1$, we say that a subgraph G' of G^* is a *t-spanner* of G^* if $V(G') = V(G^*)$, and, for every pair v, v' of vertices of G^*, if we denote by $\mathrm{dist}_{G^*}(v, v')$ the length of the shortest v-v' path in G^*, and we define $\mathrm{dist}_{G'}(v, v')$ similarly for G', then $\mathrm{dist}_{G'}(v, v') \leq t \cdot \mathrm{dist}_{G^*}(v, v')$.

Consider now any subgraph H' of H. We use the following algorithm of [ADD+93], whose goal is to construct a q-spanner H'' of H' that contains few edges. The algorithm starts with graph H'', whose vertex set is $V(H'') = V(H')$, and edge set is empty. It then processes every edge $e \in E(H')$ one by one. If graph $H'' \cup \{e\}$ contains a cycle of length at most q, then we continue to the next iteration; otherwise, we add e to H'', and continue to the next iteration. Consider the final graph H'' that is obtained at the end of the algorithm, once very edge of H' is processed. It is immediate to see that the girth of H'' is greater than q. One can also show that the resulting graph H'' is a q-spanner of H', but we do not need to use this fact. We use the following theorem from [Bol04].

Theorem 5.9 (Theorem 3.7 from [Bol04])**.** *Let G be an n-vertex graph with girth greater than q, for any integer $q > 1$. Then $|E(G)| \leq n \cdot \left\lceil n^{2/(q-2)} \right\rceil$.*

From the above theorem, $|E(H'')| \leq |V(H')|^{1+O(1/q)} = O(|V(H')|)$, as $|V(H')| \leq N$ and $q = \lceil \log N \rceil$.

We denote by $W' \subseteq G$ the origin graph of H', and we denote by $W'' \subseteq G$ the origin graph of H''. Note that $|E(W'')| \leq |E(H'')| \leq O(|V(H')|)$, and so $|V(W'')| \leq O(|E(W'')|) \leq O(|V(H')|)$. We next show the following observation.

Observation 5.10. *If Event $\mathcal{E}^b$ did not happen, then $V(W') = V(W'')$.*

Notice that the observation implies that $|V(W')| \leq O(|V(H')|)$, completing the proof of Claim 5.8. It now remains to prove Observation 5.10.

Proof of Observation 5.10. Consider any edge $e \in E(H') \setminus E(H'')$. From the construction of graph H'', there must be a simple cycle C in graph $H'' \cup \{e\}$, whose length is at most q. Consider now the subgraph W_C of G, induced by the edges of $\{R(e') \mid e' \in E(C)\}$; in other words, $W_C = R(C)$. Since the event $\mathcal{E}^b$ did not happen, from Observation 5.7, graph W_C is an even-degree graph. Therefore, if we denote by $\hat{e} = R(e)$ the origin-edge of e, then graph $W_C \setminus \{\hat{e}\}$ contains exactly two odd-degree vertices, that serve as endpoints of edge $\hat{e}$ in G. Notice however that $W_C \setminus \{\hat{e}\} \subseteq W''$. Therefore, for every edge $e \in E(H') \setminus E(H'')$, the endpoints of the origin edge $R(e)$ lie in W''. It then follows that $W' = W''$. □ □

Bad Event $\mathcal{E}$. We say that the bad event $\mathcal{E}$ happens if either bad event $\mathcal{E}^b$ happens, or bad event $\mathcal{E}^g$ does not happen. By using the Union bound, together with Claim 5.3 and Observation 5.5, we get that $\mathbf{Pr}[\mathcal{E}] \leq 0.1$.

We are now ready to complete the proof of Theorem 5.1.

5.2 Completing the Reduction from Densest k-Subgraph to Dense k-Coloring

Let $\mathrm{DkS}(G,k)$ be an input instance of the Densest k-Subgraph problem. Denote $V(G) = \{v_1, \ldots, v_N\}$. We start by constructing the auxiliary graph H, from instance $\mathrm{DkS}(G,k)$. We add isolated vertices to graph H, until $|V(H)|$ becomes an integral multiple of k, and we denote $|V(H)| = n$. Clearly, $n \leq N^{O(\log N)}$. We then consider instance $\mathrm{DkC}(H,k)$ of the Dense k-Coloring problem, where the parameter k remains unchanged. Note that, if Event $\mathcal{E}$ did not happen, then, from the definition of Events $\mathcal{E}$ and $\mathcal{E}^g$, there is a collection $\{H_1, \ldots, H_r\}$ of $r = \left\lfloor \frac{M}{k \log k} \right\rfloor$ disjoint subsets of vertices of H, such that for all $1 \leq j \leq r$, $|V(H_j)| \leq k$ holds, and additionally, $\sum_{1 \leq j \leq r} |E(H_j)| \geq 0.1 \cdot \left\lfloor \frac{M}{k \log k} \right\rfloor \cdot \mathrm{OPT}_{\mathrm{DkS}}(G,k)$. We can then define a solution $(S_1, \ldots, S_{n/k})$ to instance $\mathrm{DkC}(H,k)$ of the Dense k-Coloring problem, as follows. For all $1 \leq i \leq r$, we initially set $S_i = V(H_i)$, and for all $r < i \leq n/k$, we set $S_i = \emptyset$. Let $U = V(H) \setminus (\bigcup_{i=1}^{r} V(H_i))$. Next, we partition the vertices of U by adding them to sets $S_1, \ldots, S_{n/r}$ arbitrarily, to ensure that the cardinality of each set is exactly k. From the above discussion, if Event $\mathcal{E}$ did not happen, then:

$$\sum_{i=1}^{n/k} |E_H(S_i)| \geq \sum_{i=1}^{r} |E(H_i)| \geq 0.1 \cdot \left\lfloor \frac{M}{k \log k} \right\rfloor \cdot \mathrm{OPT}_{\mathrm{DkS}}(G,k) \geq \Omega\left(\frac{n}{k \log k}\right) \cdot \mathrm{OPT}_{\mathrm{DkS}}(G,k).$$

We conclude that, if Event $\mathcal{E}$ did not happen, then $\mathrm{OPT}_{\mathrm{DkC}}(H,k) \geq \Omega\left(\frac{n}{k \log k}\right) \cdot \mathrm{OPT}_{\mathrm{DkS}}(G,k)$.

We apply the $\alpha(n)$-approximation algorithm for Dense k-Coloring to instance $\mathrm{DkC}(H,k)$, and we denote the resulting solution by $(U_1, \ldots, U_{n/k})$. Note that:

$$\sum_{i=1}^{n/k} |E_H(U_i)| \geq \frac{\mathrm{OPT}_{\mathrm{DkC}}(H,k)}{\alpha(n)} \geq \Omega\left(\frac{n}{k \cdot \alpha(n) \cdot \log k}\right) \cdot \mathrm{OPT}_{\mathrm{DkS}}(G,k)$$

must hold. We let $U \in \{U_1, \ldots, U_{n/k}\}$ be a subset maximizing $|E_H(U_i)|$, so that

$$|E_H(U)| \geq \frac{\sum_{i=1}^{n/k} |E_H(U_i)|}{n/k} \geq \Omega\left(\frac{\mathrm{OPT}_{\mathrm{DkS}}(G,k)}{\alpha(n)\cdot \log k}\right).$$

Let $H' = H[U]$, and let $W = R(H')$ be the origin graph of H'. From Claim 5.8, if Event $\mathcal{E}$ did not happen, then $|V(W)| \leq c^* \cdot |V(H')| \leq c^* k$, for some universal constant c^*.

Lastly, we apply the algorithm from Lemma 2.3 to graph W, to obtain a subgraph W' of W with $|V(W')| = k$ and $|E(W')| \geq \Omega(|E(W)|)$. We then return $S = |V(W')|$ as the solution to the input instance $\mathrm{DkS}(G,k)$ of the Densest k-Subgraph problem. From the above discussion, $|E_G(S)| \geq \Omega(|E(W)|) \geq \Omega\left(\frac{\mathrm{OPT}_{\mathrm{DkS}}(G,k)}{\alpha(n)\cdot\log k}\right) \geq \Omega\left(\frac{\mathrm{OPT}_{\mathrm{DkS}}(G,k)}{\alpha\left(N^{O(\log N)}\right)\cdot\log N}\right)$. Therefore, if the event $\mathcal{E}$ does not happen, we obtain an $O(\alpha(N^{O(\log N)})\cdot\log N)$-approximate solution to the input instance of the Densest k-Subgraph problem. Recall that the probability of Event $\mathcal{E}$ happening is at most 0.1. Lastly, since $|V(H)| \leq N^{O(\log N)}$, it is easy to verify that the running time of the algorithm is at most $N^{O(\log N)}$.

5.3 Completing the Reduction from Densest k-Subgraph to (r,h)-Graph Partitioning

Let $\mathrm{DkS}(G,k)$ be an input instance of the Densest k-Subgraph problem with $|V(G)| = N$. Our algorithm requires the knowledge of an estimate h on the value of $\mathrm{OPT}_{\mathrm{DkS}}(G,k)$, with $h/2 \leq \mathrm{OPT}_{\mathrm{DkS}}(G,k) \leq h$. In order to overcome this difficulty, we run the algorithm for every value of h that is an integral power of 2 between 1 and $|E(G)|$, and output the best resulting solution. Therefore, it is now enough to provide a randomized algorithm that, given an estimate h with $h/2 \leq \mathrm{OPT}_{\mathrm{DkS}}(G,k) \leq h$, with a constant probability produces a solution to instance $\mathrm{DkS}(G,k)$ of Densest k-Subgraph whose value is at least $\Omega\left(\frac{\mathrm{OPT}_{\mathrm{DkS}}(G,k)}{(\alpha(N^{O(\log N)}))^3\cdot\log^2 N}\right)$, such that the running time of the algorithm is $N^{O(\log N)}$. From now on we assume that we are given an integer h with $h/2 \leq \mathrm{OPT}_{\mathrm{DkS}}(G,k) \leq h$.

As before, we denote $V(G) = \{v_0, \ldots, v_{N-1}\}$, and we construct the auxiliary graph H from instance $\mathrm{DkS}(G,k)$ of Densest k-Subgraph. We denote $|V(H)| = n$, so $n \leq N^{O(\log N)}$ holds. We then consider instance $\mathrm{GP}(H,r,h)$ of the (r,h)-Graph Partitioning problem, where $r = \left\lfloor \frac{n}{k\log k} \right\rfloor$.

Note that, if Event $\mathcal{E}$ did not happen, then, from the definition of Events $\mathcal{E}$ and $\mathcal{E}^g$, there is a collection $\{H_1, \ldots, H_r\}$ of $r = \left\lfloor \frac{n}{k\log k} \right\rfloor$ disjoint subgraphs of H, such that for all $1 \leq j \leq r$, $|E(H_j)| \leq \mathrm{OPT}_{\mathrm{DkS}}(G,k) \leq h$ holds, and $\sum_{1\leq j\leq r} |E(H_j)| \geq 0.1\cdot\left\lfloor \frac{n}{k\log k} \right\rfloor \cdot \mathrm{OPT}_{\mathrm{DkS}}(G,k)$. Therefore, we obtain a solution $(H_1, \ldots, H_r)$ to instance $\mathrm{GP}(H,r,h)$ of (r,h)-Graph Partitioning, whose value is at least $0.1 \cdot \left\lfloor \frac{n}{k\log k} \right\rfloor \cdot \mathrm{OPT}_{\mathrm{DkS}}(G,k)$. We conclude that, if Event $\mathcal{E}$ did not happen, then $\mathrm{OPT}_{\mathrm{GP}}(H,r,h) \geq \Omega\left(\frac{n}{k\log k}\right)\cdot \mathrm{OPT}_{\mathrm{DkS}}(G,k)$.

We apply the $\alpha(n)$-approximation algorithm to instance $\mathrm{GP}(H,r,h)$ of (r,h)-Graph Partitioning, obtaining a solution $(H'_1, \ldots, H'_r)$, whose value is at least $\frac{\mathrm{OPT}_{\mathrm{GP}}(H,r,h)}{\alpha(n)}$. Denote $\mathcal{H} = \{H'_1, \ldots, H'_r\}$. We partition set $\mathcal{H}$ into two subsets: set $\mathcal{H}'$ containing all graphs $H'_j \in \mathcal{H}$ with $|V(H')| \leq 100 \cdot$

$\alpha(n)k\log k$, and set $\mathcal{H}''$ containing all remaining graphs.

Assume for now that Event $\mathcal{E}$ did not happen. Then, as observed above:

$$\sum_{H'_j\in\mathcal{H}}|E(H'_j)|\geq\frac{\mathrm{OPT}_{\mathrm{GP}}(H,r,h)}{\alpha(n)}\geq\frac{n\cdot\mathrm{OPT}_{\mathrm{DkS}}(G,k)}{20k\cdot\alpha(n)\cdot\log k}.$$

Clearly, $|\mathcal{H}''|\leq\frac{n}{100\cdot\alpha(n)\cdot k\log k}$, and so:

$$\sum_{H'_j\in\mathcal{H}''}|E(H'_j)|\leq\frac{n\cdot h}{100\cdot\alpha(n)\cdot k\log k}\leq\frac{n\cdot\mathrm{OPT}_{\mathrm{DkS}}(G,k)}{100\cdot\alpha(n)\cdot k\log k}.$$

Altogether, we get that, if Event $\mathcal{E}$ did not happen, then:

$$\sum_{H'_j\in\mathcal{H}'}|E(H)|\geq\frac{n\cdot\mathrm{OPT}_{\mathrm{DkS}}(G,k)}{50k\cdot\alpha(n)\cdot\log k}.$$

We let $H^*\in\mathcal{H}'$ be the graph maximizing the number of edges. Since $|\mathcal{H}'|\leq|\mathcal{H}|=r=\left\lfloor\frac{n}{k\log k}\right\rfloor$, from the above discussion, if Event $\mathcal{E}$ did not happen, then:

$$|E(H^*)|\geq\frac{n\cdot\mathrm{OPT}_{\mathrm{DkS}}(G,k)}{r\cdot50k\cdot\alpha(n)\cdot\log k}\geq\frac{\mathrm{OPT}_{\mathrm{DkS}}(G,k)}{50\alpha(n)}.$$

From the definition of the collection $\mathcal{H}'$ of graphs, $|V(H^*)|\leq100\cdot\alpha(n)k\log k$.

Let $W=R(H^*)$ be the origin graph of H^*. From Claim 5.8, if Event $\mathcal{E}$ did not happen, then $|V(W)|\leq c^*\cdot|V(H^*)|\leq O(\alpha(n)k\log k)$.

Lastly, we apply the algorithm from Lemma 2.3 to graph W, to obtain a subgraph W' of W with $|V(W')|\leq k$ and $|E(W')|\geq\Omega\left(\frac{|E(W)|}{(\alpha(n))^2\cdot\log^2k}\right)$. We then return $S=|V(W')|$ as the solution to the input instance $\mathrm{DkS}(G,k)$ of the Densest k-Subgraph problem. From the above discussion, $|V(S)|\leq k$, and, if Event $\mathcal{E}$ did not happen, then:

$$\begin{aligned}|E_G(S)|&\geq\Omega\left(\frac{|E(W)|}{(\alpha(n))^2\cdot\log^2k}\right)\\&\geq\Omega\left(\frac{|E(H^*)|}{(\alpha(n))^2\cdot\log^2k}\right)\\&\geq\Omega\left(\frac{\mathrm{OPT}_{\mathrm{DkS}}(G,k)}{(\alpha(n))^3\cdot\log^2k}\right)\\&\geq\Omega\left(\frac{\mathrm{OPT}_{\mathrm{DkS}}(G,k)}{(\alpha(N^{O(\log N)}))^3\cdot\log^2N}\right).\end{aligned}$$

Therefore, if the event $\mathcal{E}$ does not happen, we obtain an $O((\alpha(N^{O(\log N)}))^3\cdot\log^2N)$-approximate solution to the input instance of the Densest k-Subgraph problem. Recall that the probability of

Event $\mathcal{E}$ happening is at most 0.1. Lastly, since $|V(H)| \leq N^{O(\log N)}$, it is easy to verify that the running time of the algorithm is at most $N^{O(\log N)}$.

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

In this section we establish a connection between the (r,h)-Graph Partitioning and Maximum Bounded-Crossing Subgraph problems, by proving the following two theorems.

Theorem 6.1. *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 $O(\alpha(N) \cdot \operatorname{poly}\log N)$-approximation algorithm for* Maximum Bounded-Crossing Subgraph*, where N is the number of vertices in the input instance of* Maximum Bounded-Crossing Subgraph.

Theorem 6.2. *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 $O((\alpha(n))^2 \cdot \operatorname{poly}\log n)$-approximation algorithm for* (r,h)-Graph Partitioning*, where n is the number of vertices in the input instance of* (r,h)-Graph Partitioning.

By combining Theorem 6.2 with Corollary 5.2, we obtain the following corollary.

Corollary 6.3. *Assume that Conjecture 2 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* Maximum Bounded-Crossing Subgraph.

Proof: Assume that Conjecture 2 holds and that NP $\not\subseteq$ DTIME$(n^{O(\log n)})$. Then, from Corollary 5.2, for some constant $0 < \varepsilon \leq 1/2$, there is no efficient factor-$2^{(\log n)^{\varepsilon}}$-approximation algorithm for (r,h)-Graph Partitioning, where n is the number of vertices in the input graph.

We let $\varepsilon' = \varepsilon/c$, where c is a sufficiently large constant. We now prove that there is no efficient $2^{(\log n)^{\varepsilon'}}$-approximation algorithm for Maximum Bounded-Crossing Subgraph. Indeed, assume for contradiction that there is an efficient $2^{(\log n)^{\varepsilon'}}$-approximation algorithm $\mathcal{A}$ for Maximum Bounded-Crossing Subgraph. From Theorem 6.2, there is an efficient $c' \cdot (\alpha(N))^2 \cdot \operatorname{poly}\log N$-approximation algorithm for (r,h)-Graph Partitioning, where N is the number of vertices in the input instance of (r,h)-Graph Partitioning, c' is some constant, and $\alpha(N) = 2^{(\log N)^{\varepsilon'}}$. Notice however that $c' \cdot (\alpha(N))^2 \cdot \operatorname{poly}\log N \leq 2^{(\log N)^{\varepsilon}}$ holds, if the constant c is large enough.

Therefore, we obtain an efficient factor-$2^{(\log n)^{\varepsilon}}$-approximation algorithm for (r,h)-Graph Partitioning, a contradiction. □

In the remainder of this section, we prove Theorems 6.2 and 5.2. We start by proving two auxiliary lemmas that will be used in the proofs of both theorems. We then complete the proofs of Theorem 6.1 and Theorem 6.2 in sections Section 6.2 and Section 6.3, respectively.

6.1 Auxiliary Lemmas

We start with the following definition, that will be used in the proofs of both theorems.

Definition 6.4. *Let* $\mathrm{GP}(G, r, h)$ *be an instance of* (r,h)-Graph Partitioning*, and let* $\{H_1, \ldots, H_r\}$ *be a solution to this instance. We say that this solution is* good*, if for all* $1 \leq i \leq r$, $h/2 \leq |E(H_i)| \leq h$.

We are now ready to state the first auxiliary lemma.

Lemma 6.5. *There is an efficient algorithm, that, given a graph* G *with* $|V(G)| = n$*, integers* $r, h > 0$ *and any solution* $\mathcal{H}$ *to instance* $\mathrm{GP}(G, r, h)$ *of* (r,h)-Graph Partitioning*, computes positive integers* $r^* \leq r, h^* \leq h$ *and a subset* $\mathcal{H}^* \subseteq \mathcal{H}$ *of subgraphs of* G*, such that* $\mathcal{H}^*$ *is a good solution to instance* $\mathrm{GP}(G, r^*, h^*)$*, of* (r,h)-Graph Partitioning*, and* $\sum_{H \in \mathcal{H}^*} |E(H)| \geq \frac{\sum_{H' \in \mathcal{H}} |E(H')|}{4 \log n}$.

Proof: The idea of the proof is to partition the graphs in $\mathcal{H}$ geometrically into groups by the cardinalities of their edge sets, and then select a group maximizing the total number of edges in its subgraphs.

Specifically, let $q = \lceil 2 \log n \rceil$. For all $1 \leq i \leq q$, we let $\mathcal{H}_i \subseteq \mathcal{H}$ contain all graphs H with $2^{i-1} \leq |E(H)| < 2^i$. It is easy to verify that $\mathcal{H}_1 \ldots, \mathcal{H}_q$ partition $\mathcal{H}$. Clearly, there must be an index $1 \leq i^* \leq q$, with $\sum_{H \in \mathcal{H}_{i^*}} |E(H)| \geq \frac{\sum_{H' \in \mathcal{H}} |E(H')|}{q} \geq \frac{\sum_{H' \in \mathcal{H}} |E(H')|}{4 \log n}$. We set $r^* = |\mathcal{H}_{i^*}|$, $h^* = 2^{i^*}$, and we let $\mathcal{H}^* = \mathcal{H}_{i^*}$. It is immediate to verify that $\mathcal{H}^*$ is a good solution to instance $\mathrm{GP}(G, r^*, h^*)$, of (r,h)-Graph Partitioning, and, from the above discussion, $\sum_{H \in \mathcal{H}^*} |E(H)| \geq \frac{\sum_{H' \in \mathcal{H}} |E(H')|}{4 \log n}$. □

We are now ready to prove our second auxiliary lemma.

Lemma 6.6. *There is an efficient algorithm, whose input consists of an instance* $\mathrm{MBCS}(G, L)$ *of the* Maximum Bounded-Crossing Subgraph *problem with* $|V(G)| = N$*, where* N *is greater than a sufficiently large constant, together with a solution* H *to this instance, such that* $|E(H)| \geq 4N \log^6 N$ *holds. The algorithm computes integers* $r, h > 0$*, such that* $r \cdot h^2 \leq L \cdot \log^6 N$ *and* $r \cdot h \geq \Omega(|E(H)|/\log N)$ *hold, together with a good solution* $\{H_1, \ldots, H_r\}$ *to instance* $\mathrm{GP}(G, r, h)$ *of* (r,h)-Graph Partitioning*.*

Proof: Let $\hat{G}$ be any graph. A *cut* in $\hat{G}$ is a partition (A, B) of vertices of $\hat{G}$ into two non-empty subsets. The *value* of the cut (A, B) is $|E(A, B)|$. For a parameter $1/2 < \beta < 1$, we say that cut (A, B) is β*-balanced*, if $|E(A)|, |E(B)| \leq \beta \cdot |E(\hat{G})|$. We say that cut (A, B) is a *minimum* β*-balanced cut* if it is a β-balanced cut whose value is the smallest among all such cuts. We use the following theorem that follows from the results of [ARV09], and was formally proved in [CT22].

Theorem 6.7 (Theorem 4.11 in the full version of [CT22])**.** *There is an efficient algorithm, that, given a graph* $\hat{G}$ *with* $|V(\hat{G})| = \hat{N}$*, computes a* γ*-balanced cut in* $\hat{G}$*, whose value is at most* $O(\sqrt{\log \hat{N}})$ *times the value of the minimum* $(3/4)$*-balanced cut in* $\hat{G}$*, for some universal constant* $3/4 < \gamma < 1$ *that does not depend on* $\hat{N}$*.*

We use the following theorem, that is a simple corollary of the Planar Separator Theorem by Lipton and Tarjan [LT79], and was formally proved in [CT22]. A variation of this theorem for vertex-balanced cuts was proved in [PSS96].

Theorem 6.8 (Lemma 4.12 in the full version of [CT22]). *Let $\hat{G}$ be a connected graph with m edges and maximum vertex degree $\Delta < \frac{m}{2^{40}}$. If $\mathsf{CrN}(\hat{G}) \leq \frac{m^2}{2^{40}}$, then the value of the minimum $(3/4)$-balanced cut in $\hat{G}$ is at most $O\left(\sqrt{\mathsf{CrN}(\hat{G}) + \Delta \cdot m}\right)$.*

We are now ready to complete the proof of Lemma 6.6. Recall that we are given an instance $\mathrm{MBCS}(G, L)$ of Maximum Bounded-Crossing Subgraph, where $|V(G)| = N$, together with a solution H to this instance, such that $|E(H)| \geq 4N \log^6 N$.

The algorithm starts by iteratively decomposing graph H into smaller subgraphs. Throughout the decomposition procedure, we maintain a collection $\mathcal{H}$ of connected subgraphs of H, that are all mutually disjoint. Each graph $H' \in \mathcal{H}$ is marked as either *active* or *inactive*. At the beginning of the algorithm, we let $\mathcal{H}$ contain all connected components of H, which are all marked as active. The algorithm performs iterations, as long as at least one graph in $\mathcal{H}$ is inactive.

In order to execute an iteration, we select an arbitrary active graph $H' \in \mathcal{H}$. We apply the algorithm from Theorem 6.7 to compute a γ-balanced cut (A, B) of H'. If $|E_{H'}(A, B)| \geq \frac{|E(H')|}{\log^2 N}$, then we mark H' as inactive and continue to the next iteration. Otherwise, we remove graph H' from $\mathcal{H}$, and we add all connected components of graphs $H'[A]$ and $H'[B]$ to $\mathcal{H}$, that are all marked as active graphs. We then continue to the next iteration. This completes the description of the decomposition procedure. Let $\mathcal{H}'$ be the collection $\mathcal{H}$ of subgraphs of H that we obtain at the end of the procedure. We prove the following simple observation.

Observation 6.9. $\sum_{H' \in \mathcal{H}'} |E(H')| \geq |E(H)|/2$.

Proof: We use a charging scheme. We observe the set $\mathcal{H}$ of graphs over the course of the partitioning procedure. Throughout the execution of the partitioning procedure, we denote by $E' = E(H) \setminus \left(\bigcup_{H' \in \mathcal{H}} E(H')\right)$, and we call the edges of E' *deleted edges*. Over the course of the partitioning procedure we maintain, for every edge $e \in E(H)$, a non-negative value $c(e)$, that we refer to as the *charge* of e. We will ensure that, at every point of the algorithm's execution, $\sum_{e \in E(H)} c(e) \geq |E'|$, and that, for every edge $e \in E(H)$, $c(e) \leq 1/2$ always holds. We note that, even when an edge e is added to the set E' of deleted edges, its charge $c(e)$ may remain strictly positive. It is then easy to verify that, at the end of the algorithm, $|E'| \leq \sum_{e \in E(H)} c(e) \leq |E(H)|/2$ holds, and so $\sum_{H' \in \mathcal{H}'} |E(H')| \geq |E(H)|/2$.

It now remains to describe the assignment of the charge values $c(e)$ to the edges $e \in E(H)$, for which the above properties hold. Initially, $E' = \emptyset$, and we set $c(e) = 0$ for every edge $e \in E(H)$.

Consider now some iteration of the algorithm, and assume that, at the beginning of the iteration, $\sum_{e \in E(H)} c(e) \geq |E'|$ holds. Let $H' \in \mathcal{H}$ be the graph that was processed in the current iteration, and let (A, B) be the cut in H' that the algorithm computed. If $|E_{H'}(A, B)| \geq \frac{|E(H')|}{\log^2 N}$, then no new edges were added to E' in the current iteration, and the charge values $c(e)$ remain unchanged for all edges $e \in E(H)$. Assume now that $|E_{H'}(A, B)| < \frac{|E(H')|}{\log^2 N}$ holds, and denote $E'' = E_{H'}(A, B)$. Then in the current iteration, the edges of E'' were added to set E'. We increase the charge $c(e)$ of every edge $e \in E(H')$ by $\frac{|E''|}{|E(H')|}$, and leave all other edge charges unchanged. This ensures that $\sum_{e \in E(H)} c(e) \geq |E'|$ holds at the end of the iteration. Since $|E''| < \frac{|E(H')|}{\log^2 N}$, for every edge $e \in E(H')$, the charge $c(e)$ increases by at most $\frac{1}{\log^2 N}$ in the current iteration.

From the above discussion, at the end of the algorithm, $|E'| \leq \sum_{e \in E(H)} c(e)$ holds. It now remains to show that for every edge $e \in E(H)$, $c(e) \leq 1/2$ holds at the end of the algorithm.

Consider any edge $e \in E(H)$, and denote by $H_1, H_2, \ldots, H_r$ the sequence of subgraphs of H that belonged to $\mathcal{H}$ over the course of the algorithm, and contained e. In other words, $H_1 = H$, and, for all $1 < i \leq r$, graph H_i was obtained via a balanced cut from graph H_{i-1}. Then the charge of e has increased in at most $r+1$ iterations, and in each such iteration, the increase in the charge was bounded by $\frac{1}{\log^2 N}$. Furthermore, for all $1 < i \leq r$, $|E(H_i)| \leq \gamma \cdot |E(H_{i+1})|$ holds, and so $r \leq O(\log N)$ as γ is a constant. Therefore, at the end of the algorithm, $c(e) \leq \frac{r+1}{\log^2 N} \leq \frac{1}{2}$, since we have assumed that N is sufficiently large. □

Consider now the final collection $\mathcal{H}'$ of graphs. We say that a graph $H' \in \mathcal{H}'$ is *dense* iff $|E(H')| \geq |V(H')| \cdot \log^6 N$, and otherwise we say it is *sparse*. We partition the set $\mathcal{H}'$ of graphs into a collection $\mathcal{H}^d$ containing all dense graphs and a collection $\mathcal{H}^s$ containing all sparse graphs. We need the following simple observation.

Observation 6.10. $\sum_{H' \in \mathcal{H}^d} |E(H')| \geq \frac{|E(H)|}{4} \geq N \log^6 N$.

Proof: Assume for contradiction that $\sum_{H' \in \mathcal{H}^d} |E(H')| < \frac{|E(H)|}{4}$. Since, from Observation 6.9 $\sum_{H' \in \mathcal{H}'} |E(H')| \geq |E(H)|/2$, we get that $\sum_{H' \in \mathcal{H}^s} |E(H')| > \frac{|E(H)|}{4}$. However:

$$\sum_{H' \in \mathcal{H}^s} |E(H')| \leq \sum_{H' \in \mathcal{H}^s} |V(H')| \cdot \log^6 N \leq N \log^6 N.$$

We then conclude that $|E(H)| < 4 \sum_{H' \in \mathcal{H}^s} |E(H')| \leq 4N \log^6 N$, contradicting the statement of Lemma 6.6. □

We also need the following obsevation.

Observation 6.11. *Let $H' \in \mathcal{H}^d$ be a dense graph. Then* $\mathsf{CrN}(H') \geq \Omega\left(\frac{|E(H')|^2}{\log^5 N}\right)$.

We provide the proof of Observation 6.11 below, after we complete the proof of Lemma 6.6 using it. Let $r' = |\mathcal{H}^d|$ and $h' = \max_{H' \in \mathcal{H}^d}\{|E(H')|\}$. Clearly, collection $\mathcal{H}^d$ of graphs is a valid solution to instance $\mathrm{GP}(G, r', h')$ of (r,h)-Graph Partitioning. We apply the algorithm from Lemma 6.5 to the soluton $\mathcal{H}^d$ to instance $\mathrm{GP}(G, r', h')$ of (r,h)-Graph Partitioning. Recall that the algorithm computes positive integers $r \leq r', h \leq h'$, and a subset $\mathcal{H}^* \subseteq \mathcal{H}^d$ of subgraphs of G, such that $\mathcal{H}^*$ is a good solution to instance $\mathrm{GP}(G, r, h)$, of (r,h)-Graph Partitioning, and $\sum_{H' \in \mathcal{H}^*} |E(H')| \geq \frac{\sum_{H' \in \mathcal{H}^d} |E(H')|}{4 \log N}$. It now remains to verify that $r \cdot h^2 \leq L \cdot \log^6 N$ and $r \cdot h \geq \Omega(|E(H)|/\log N)$ hold.

Observe first that:

$$r \cdot h \geq \sum_{H' \in \mathcal{H}^*} |E(H')| \geq \frac{\sum_{H' \in \mathcal{H}^d} |E(H')|}{4 \log N} \geq \frac{|E(H)|}{16 \log N},$$

from Observation 6.10.

Finally, since for each graph $H' \in \mathcal{H}^d$, $\mathsf{CrN}(H') \geq \Omega\left(\frac{|E(H')|^2}{\log^5 N}\right)$ holds, and since, for every graph $H' \in \mathcal{H}^*$, $|E(H')| \geq h/2$ holds, we get that:

$$r \cdot h^2 \leq \sum_{H' \in \mathcal{H}^*} 4 \cdot |E(H')|^2 \leq \sum_{H' \in \mathcal{H}^*} O(\log^5 N) \cdot \mathsf{CrN}(H') \leq O(\log^5 N) \cdot \mathsf{CrN}(H) \leq L \cdot \log^6 N,$$

since N is large enough.

In order to complete the proof of Lemma 6.6, it is now enough to prove Observation 6.11, which we do next.

Proof of Observation 6.11. Since graph H' is marked inactive by the algorithm, the γ-balanced cut of H' computed by the algorithm from Theorem 6.7 had value at least $\frac{|E(H')|}{\log^2 N}$. Therefore the minimum (3/4)-balanced cut of H' has value at least $\Omega\left(\frac{|E(H')|}{\log^{2.5} N}\right)$.

Let Δ denote the maximum vertex degree in H'. Clearly, $\Delta \leq V(H')$ must hold. On the other hand, from the definition of a dense graph, $|E(H')| \geq |V(H')| \cdot \log^6 N$, and so $\Delta \leq |V(H')| \leq \frac{|E(H')|}{\log^6 N} < \frac{|E(H')|}{2^{40}}$, if N is sufficiently large.

Recall that, from Theorem 6.8, either $\mathsf{CrN}(H') \geq \frac{|E(H')|^2}{2^{40}}$, or the value of minimum (3/4)-balanced cut in H' is at most $\sqrt{\mathsf{CrN}(H') + \Delta \cdot |E(H')|}$. In the former case, we immediately get that $\mathsf{CrN}(H') \geq \Omega\left(\frac{|E(H')|^2}{\log^5 N}\right)$. In the latter case, since the value of the minimum (3/4)-balanced cut in H' is $\Omega\left(\frac{|E(H')|}{\log^{2.5} N}\right)$, we get that $\sqrt{\mathsf{CrN}(H') + \Delta \cdot |E(H')|} \geq \Omega\left(\frac{|E(H')|}{\log^{2.5} N}\right)$. Moreover, since $\Delta \leq \frac{|E(H')|}{\log^6 N}$:

$$\mathsf{CrN}(H') \geq \Omega\left(\frac{|E(H')|^2}{\log^5 N}\right) - \Delta \cdot |E(H')| \geq \Omega\left(\frac{|E(H')|^2}{\log^5 N}\right) - \frac{|E(H')|^2}{\log^6 N} \geq \Omega\left(\frac{|E(H')|^2}{\log^5 N}\right),$$

since N is sufficiently large. □ □

6.2 Reduction from Maximum Bounded-Crossing Subgraph to (r,h)-Graph Partitioning: Proof of Theorem 6.1

In this subsection we prove Theorem 6.1. Let MBCS(G, L) be a given instance of Maximum Bounded-Crossing Subgraph, with $|V(G)| = N$. Note that we can assume without loss of generality that G contains no isolated vertices, since all such vertices can be deleted without changing the problem.

As our first step, we compute an arbitrary spanning forest F of graph G. Since G contains no isolated vertices, $|E(F)| \geq N/2$. Clearly, $\mathsf{CrN}(F) = 0$. Consider now the optimal solution H^* to instance MBCS(G, L) of Maximum Bounded-Crossing Subgraph. If $|E(H^*)| < 4N \cdot \log^6 N$, then F is a factor-$O(\log^6 N)$ approximate solution to instance MBCS(G, L).

Assume now that $|E(H^*)| \geq 4N \log^6 N$. Then, from Lemma 6.6, there exist integers $r, h > 0$ with $r \cdot h^2 \leq L \cdot \log^6 N$ and $r \cdot h \geq \Omega(|E(H^*)|/\log N)$, such that there exists a good solution to instance GP(G, r, h) of (r,h)-Graph Partitioning. We will now attempt to guess such integers r, h, and then use the approximation algorithm for the (r,h)-Graph Partitioning problem, in order to compute a solution to the corresponding instance GP(G, r, h) of (r,h)-Graph Partitioning, whose value is sufficiently high.

We say that a pair (r,h) of positive integers is *eligible*, if $r\cdot h^2\leq L\cdot\log^6 N$. For each eligible pair (r,h) of integers, we consider the instance $\mathrm{GP}(G,r,h)$ of (r,h)-Graph Partitioning and we use the $\alpha(n)$-approximation algorithm for the (r,h)-Graph Partitioning problem to compute a solution $\mathcal{H}_{r,h}$ to the instance $\mathrm{GP}(G,r,h)$, such that $\sum_{H'\in\mathcal{H}_{r,h}}|E(H')|\geq\frac{\mathrm{OPT}_{\mathrm{GP}}(G,r,h)}{\alpha(N)}$ (since $|V(G)|=N$). We use the following observation.

Observation 6.12. *If $\mathrm{OPT}_{\mathrm{MBCS}}(G,L)\geq 4N\log^6 N$, then there exists an eligible pair (r,h) of integers, with:*

$$\sum_{H'\in\mathcal{H}_{r,h}}|E(H')|\geq\Omega\left(\frac{\mathrm{OPT}_{\mathrm{MBCS}}(G,L)}{\alpha(N)\log N}\right).$$

Proof: Let H^* be an optimal solution to the instance $\mathrm{MBCS}(G,L)$. From Lemma 6.6, there is an eligible pair (r^*,h^*) of integers, with $r^*\cdot h^*\geq\Omega(E(H^*)|/\log N)$, so that there exists a good solution $\mathcal{H}^*$ to instance $\mathrm{GP}(G,r^*,h^*)$ of (r,h)-Graph Partitioning. From the definition of a good solution,

$$\sum_{H'\in\mathcal{H}^*}|E(H')|\geq\frac{r^*h^*}{2}\geq\Omega\left(\frac{|E(H^*)|}{\log N}\right)\geq\Omega\left(\frac{\mathrm{OPT}_{\mathrm{MBCS}}(G,L)}{\log N}\right).$$

Therefore, if $\mathcal{H}_{r^*,h^*}$ is the approximate solution that we obtained for instance $\mathrm{GP}(G,r^*,h^*)$ of (r,h)-Graph Partitioning, then:

$$\sum_{H'\in\mathcal{H}_{r^*,h^*}}|E(H')|\geq\Omega\left(\frac{\mathrm{OPT}_{\mathrm{GP}}(G,r^*,h^*)}{\alpha(N)}\right)\geq\Omega\left(\frac{\sum_{H'\in\mathcal{H}^*}|E(H')|}{\alpha(N)}\right)\geq\Omega\left(\frac{\mathrm{OPT}_{\mathrm{MBCS}}(G,L)}{\alpha(N)\log N}\right),$$

and the observation follows. □

Let (r',h') be the eligible pair of integers that maximizes $\sum_{H\in\mathcal{H}_{r',h'}}|E(H)|$. For every graph $H\in\mathcal{H}_{r',h'}$, let $\tilde{H}$ be a graph that is obtained from H as follows. We set $V(\tilde{H})=V(H)$, and we let $E(\tilde{H})$ contain an arbitrary subset of $\left\lfloor\frac{|E(H)|}{\log^3 N}\right\rfloor$ edges of $E(H)$. Note that:

$$\frac{|E(H)|}{2\log^3 N}\leq|E(\tilde{H})|\leq\frac{h'}{\log^3 N}.$$

Finally, we define a graph $H'=\bigcup_{H\in\mathcal{H}_{r',h'}}\tilde{H}$. Since the crossing number of any m-edge graph is bounded by m^2, it is easy to verify that:

$$\mathsf{CrN}(H')\leq\sum_{H\in\mathcal{H}_{r',h'}}\mathsf{CrN}(\tilde{H})\leq r'\cdot\frac{(h')^2}{\log^6 N}\leq L.$$

Moreover:

$$|E(H')|=\sum_{H\in\mathcal{H}_{r',h'}}|E(\tilde{H})|\geq\sum_{H\in\mathcal{H}_{r,h}}\frac{|E(H)|}{\log^3 N}\geq\Omega\left(\frac{\mathrm{OPT}_{\mathrm{MBCS}}(G,L)}{\alpha(N)\log^4 N}\right).$$

Recall that we have computed a spanning forest F of G. We return the graph in $\{F, H'\}$ that contains more edges as the outcome of the algorithm. From the above discussion, we obtain an $O(\alpha(N) \cdot \operatorname{poly}\log N)$-approximate solution.

6.3 Reduction from (r,h)-Graph Partitioning to Maximum Bounded-Crossing Subgraph– Proof of Theorem 6.2

In this subsection we prove Theorem 6.2. Let $\mathrm{GP}(G, r, h)$ be the input instance of (r,h)-Graph Partitioning, and denote $n = |V(G)|$. For convenience, we will assume that the value $C^* = \mathrm{OPT}_{\mathrm{GP}}(G, r, h)$ is known to the algorithm: since $0 \leq \mathrm{OPT}_{\mathrm{GP}}(G, r, h) \leq |E(G)|$ holds, and $\mathrm{OPT}_{\mathrm{GP}}(G, r, h)$ is an integer, we can try all possible guesses for the value C^*, and then output the best of the resulting solutions. It is sufficient to ensure that the algorithm correctly computes an $O((\alpha(n))^2 \cdot \operatorname{poly}\log n)$-approximate solution to instance $\mathrm{GP}(G, r, h)$ if the guess C^* is correct, that is, $C^* = \mathrm{OPT}_{\mathrm{GP}}(G, r, h)$. From now on we assume that we are given a value $C^* = \mathrm{OPT}_{\mathrm{GP}}(G, r, h)$.

We distinguish between two cases. The first case happens if $C^* \geq 16n\alpha(n)\log^7 n$. In this case, we proceed as follows. By applying Lemma 6.5 to the optimal solution to instance $\mathrm{GP}(G, r, h)$, we conclude that there are positive integers $r^* \leq r, h^* \leq h$ and a collectoin $\mathcal{H}^*$ of subgraphs of G, such that $\mathcal{H}^*$ is a good solution to instance $\mathrm{GP}(G, r^*, h^*)$, of (r,h)-Graph Partitioning, and $r^*h^* \geq \sum_{H \in \mathcal{H}^*} |E(H)| \geq \frac{C^*}{4\log n}$. Since the values of integers r^*, h^* are not known to our algorithm, we will try all possible candidate values $1 \leq r' \leq r$ and $1 \leq h' \leq h$ with $r'h' \geq \frac{C^*}{4\log n}$. For each such pair (r', h') of integers, we will compute a solution $\mathcal{H}_{r',h'}$ to instance $\mathrm{GP}(G, r', h')$. We will then output the best solution from among $\{\mathcal{H}_{r',h'} \mid r'h' \geq \frac{C^*}{4\log n}\}$. It is sufficient to ensure that, for integers $(r', h') = (r^*, h^*)$, the value of the resulting solution $\mathcal{H}_{r',h'}$ is close to $C^* = \mathrm{OPT}_{\mathrm{GP}}(G, r, h)$.

Consider now a pair of integers $1 \leq r' \leq r$ and $1 \leq h' \leq h$ with $r'h' \geq \frac{C^*}{4\log n}$, and assume that values r', h' were guessed correctly, that is, $(r', h') = (r^*, h^*)$. In other words, there is a good solution $\mathcal{H}^*$ to instance $\mathrm{GP}(G, r^*, h^*)$, of (r,h)-Graph Partitioning, whose value is at least $\frac{C^*}{4\log n}$. Consider now the graph $H' = \bigcup_{H \in \mathcal{H}^*} H$. Since the crossing number of a graph H may not be higher than $|E(H)|^2$, we get that $\mathsf{CrN}(H) \leq r' \cdot (h')^2$. Let $L = r' \cdot (h')^2$, and consider instance $\mathrm{MBCS}(G, L)$ of the Maximum Bounded-Crossing Subgraph problem. From the above discussion, the value of the optimal solution to this problem is at least $|E(H)| \geq \frac{\mathrm{OPT}_{\mathrm{GP}}(G,r,h)}{4\log n}$. Therefore, by applying the $\alpha(N)$-approximation algorithm for Maximum Bounded-Crossing Subgraph to this instance, we obtain a solution G' to instance $\mathrm{MBCS}(G, L)$ of Maximum Bounded-Crossing Subgraph, whose value is at least $\frac{C^*}{4\alpha(n)\log n}$. Since we have assumed that $C^* \geq 16n\alpha(n)\log^7 n$, we get that $|E(G')| \geq 4n\log^6 n$. We can now use the algorithm from Lemma 6.6 to compute integers $r'', h'' > 0$, such that $r'' \cdot (h'')^2 \leq L \cdot \log^6 n$, together with a good solution $\mathcal{H}'$ to instance $\mathrm{GP}(G, r'', h'')$, whose value is at least $\Omega\left(\frac{|E(G')|}{\log n}\right) \geq \Omega\left(\frac{C^*}{\alpha(n)\log^2 n}\right)$. Notice however that it is possible that $r'' > r'$ or $h'' > h'$ hold, so the solution that we obtain may not be a valid solution to instance $\mathrm{GP}(G, r', h')$ of (r,h)-Graph Partitioning. We show that, if $(r', h') = (r^*, h^*)$, then h'' cannot be much larger than h'. We then slightly modify solution $\mathcal{H}'$, to transform it into a valid solution $\mathcal{H}_{r',h'}$ to instance $\mathrm{GP}(G, r', h')$, while only decreasing the solution cost slightly. This completes the computation of the solution $\mathcal{H}_{r',h'}$ associated with parameters (r', h'), and the algorithm for the first case.

Consider now the second case, where $C^* < 16n\alpha(n)\log^7 n$. In this case, we start by computing a

maximal subgraph F of G, such that F is a forest, with maximum vertex degree at most h. Let S be the set of all vertices of G that are adjacent to at least one edge of F, and denote $|S| = n'$.

We consider two subcases of Case 2. The first subcase happens if $|E(F)| \geq \frac{C^*}{32\alpha(n')\cdot\log^7 n'}$. In this case, we show an algorithm that decomposes F into r subgraphs containing at most h edges each, so that the total number of edges in all such subgraphs is close to $|E(F)|$. Therefore, we obtain a solution $\mathcal{H}$ to instance $\mathrm{GP}(G,r,h)$, whose value is close to $C^* = \mathrm{OPT}_{\mathrm{GP}}(G,r,h)$. Consider now the second subcase, where $|E(F)| < \frac{C^*}{32\alpha(n')\cdot\log^7 n'}$. We show that in this case, there is a solution to instance $\mathrm{GP}(G[S],r,h)$ of (r,h)-Graph Partitioning, whose value is at least $\frac{C^*}{2}$. From now on we only consider instane $\mathrm{GP}(G[S],r,h)$. We assume again that we are given the value C^{**} of the optimal solution to this instance, where $\frac{C^*}{2} \leq C^{**} \leq C^*$. As before, this can be assumed since we can try all guesses for the value C^{**}, and it is sufficient to ensure that the algorithm works correctly if the value C^{**} is guessed correctly. Recall that we have denoted $n' = |S|$. Since $(h-1)\cdot|S| \leq |E(F)| \leq \frac{C^*}{32\alpha(n')\cdot\log^7 n'}$, while $C^{**} \geq \frac{C^*}{2}$, we get that $C^{**} \geq 16n'\cdot\alpha(n')\cdot\log^7 n'$.

We have now obtained a new instance $\mathrm{GP}(G[S],r,h)$ of (r,h)-Graph Partitioning, in which the value of the optimal solution $C^{**} \geq 16n\alpha(n)\log^7 n'$, where $n' = |S|$. We can now repeat our algorithm for Case 1, to obtain the desired approximate solution to instance $\mathrm{GP}(G[S],r,h)$, which, in turn will provide an approximate solution to the original instance $\mathrm{GP}(G,r,h)$.

We now turn to the formal proof of Theorem 6.2. We assume that we are given an instance $\mathrm{GP}(G,r,h)$ of the (r,h)-Graph Partitioning problem, where $|V(G)| = n$, together with a guess C^* on the value $\mathrm{OPT}_{\mathrm{GP}}(G,r,h)$ of the optimal solution to this instance. Our goal is to compute a solution $\mathcal{H}$ to instance $\mathrm{GP}(G,r,h)$ of (r,h)-Graph Partitioning, such that, if $C^* = \mathrm{OPT}_{\mathrm{GP}}(G,r,h)$, then the value of the solution $\mathcal{H}$ is at least $\Omega\left(\frac{C^*}{(\alpha(n))^2\mathrm{poly}\log n}\right)$. Note that we can assume that n is greater than a sufficiently large constant, since otherwise we can solve the problem efficiently via exhaustive search. We distinguish between two cases, depending on whether $C^* \geq 16n\alpha(n)\log^7 n$ holds.

6.3.1 Case 1: $C^* \geq 16n\alpha(n)\log^7 n$

Applying Lemma 6.5 to the optimal solution tp instance $\mathrm{GP}(G,r,h)$, we conclude that there are positive integers $r^* \leq r, h^* \leq h$ and a collection $\mathcal{H}^*$ of subgraphs of G, such that $\mathcal{H}^*$ is a good solution to instance $\mathrm{GP}(G,r^*,h^*)$, of (r,h)-Graph Partitioning. Moreover, if $C^* = \mathrm{OPT}_{\mathrm{GP}}(G,r,h)$, then $r^*\cdot h^* \geq \sum_{H\in\mathcal{H}^*}|E(H)| \geq \frac{C^*}{4\log n}$. Our algorithm tries all possible values of integers $1 \leq r' \leq r$ and $1 \leq h' \leq h$ with $r'h' \geq \frac{C^*}{4\log n}$. For each such pair (r',h') of integers, we will compute a solution $\mathcal{H}_{r',h'}$ to instance $\mathrm{GP}(G,r',h')$. At the end, our algorithm will output the best solution from among $\{\mathcal{H}_{r',h'} \mid r'h' \geq \frac{C^*}{4\log n}\}$. It is sufficient to ensure that, for integers $(r',h') = (r^*,h^*)$, the value of the resulting solution $\mathcal{H}_{r',h'}$ is at least $\Omega\left(\frac{C^*}{(\alpha(n))^2\mathrm{poly}\log n}\right)$.

From now on we fix a pair $1 \leq r' \leq r$, $1 \leq h' \leq h$ of integers, with $r'h' \geq \frac{C^*}{4\log n}$. Let $L = r'\cdot(h')^2$. We apply the $\alpha(N)$-approximation algorithm for the Maximum Bounded-Crossing Subgraph problem to instance $\mathrm{MBCS}(G,L)$, and obtain a solution that we denote by H. We use the following observation.

Observation 6.13. *If $C^* = \mathrm{OPT}_{\mathrm{GP}}(G,r,h)$, $r' = r^*$ and $h' = h^*$, then $|E(H)| \geq \frac{C^*}{4\alpha(n)\log n}$ must hold.*

Proof: Assume that $C^* = \mathrm{OPT}_{\mathrm{GP}}(G, r, h)$, $r' = r^*$ and $h' = h^*$. Recall that there exists a good solution $\mathcal{H}^*$ to instance $\mathrm{GP}(G, r^*, h^*)$, with $\sum_{H' \in \mathcal{H}^*} |E(H')| \geq \frac{C^*}{4 \log n}$. Consider the graph $\hat{H} = \bigcup_{H' \in \mathcal{H}^*} H'$. Since the crossing number of a graph G' may not be higher than $|E(G')|^2$, we get that $\mathsf{CrN}(\hat{H}) \leq \sum_{H' \in \mathcal{H}^*} \mathsf{CrN}(H') \leq r' \cdot (h')^2 = L$. Therefore, $\hat{H}$ is a valid solution to instance $\mathrm{MBCS}(G, L)$, whose value is at least $\frac{C^*}{4 \log n}$. Since we use an $\alpha(N)$-approximation algorithm for Maximum Bounded-Crossing Subgraph, and $|V(G)| = n$, we get that $|E(H)| \geq \frac{C^*}{4\alpha(n) \log n}$. □

If $|E(H)| < \frac{C^*}{4\alpha(n) \log n}$, then we terminate the algorithm and return an empty solution: in this case, we are guaranteed that either C^*, or r', h' are guessed incorrectly. Therefore, we assume from now on that $|E(H)| \geq \frac{C^*}{4\alpha(n) \log n}$ holds. Note that, since in Case 1, $C^* \geq 16n\alpha(n) \log^7 n$ holds, we are guaranteed that $|E(H)| \geq 4n \log^6 n$.

Next, we apply the algorithm from Lemma 6.6 to instance $\mathrm{MBCS}(G, L)$ of Maximum Bounded-Crossing Subgraph, to compute integers $r'', h'' > 0$, such that $r'' \cdot (h'')^2 \leq L \cdot \log^6 n = r' \cdot (h')^2 \cdot \log^6 n$, and $r'' \cdot h'' \geq \Omega\left(\frac{|E(H)|}{\log n}\right) \geq \Omega\left(\frac{C^*}{\alpha(n) \log^2 n}\right)$. The algorithm also computes a good solution $\mathcal{H}'$ to instance $\mathrm{GP}(G, r'', h'')$ of (r,h)-Graph Partitioning. Note that, while the value of the solution $\mathcal{H}'$ to instance $\mathrm{GP}(G, r'', h'')$ is guaranteed to be close to C^*, we are only guaranted that $\mathcal{H}'$ is a valid solution to instance $\mathrm{GP}(G, r'', h'')$ of the problem, and it may not be a valid solution to instance $\mathrm{GP}(G, r', h')$. In our next steps, we will either correctly established that at least one of C^*, r', h' was not guessed correctly; or we will slightly modify $\mathcal{H}'$ to obtain a valid solution to intance $\mathrm{GP}(G, r', h')$ of (r,h)-Graph Partitioning, whose value remains close to that of $\mathcal{H}'$. We start with the following observation.

Observation 6.14. *There is a large enough constant c', such that, if $C^* = \mathrm{OPT}_{\mathrm{GP}}(G, r, h)$, $r' = r^*$ and $h' = h^*$ hold, then $h'' \leq c' h' \cdot \alpha(n) \cdot \log^8 n$.*

Proof: Recall that we have established that:

$$r'' \cdot h'' \geq \Omega\left(\frac{C^*}{\alpha(n) \log^2 n}\right).$$

Assume for contradicton that $h'' > c' h' \cdot \alpha(n) \cdot \log^8 n$, where c' is a large enough constant. Then:

$$r'' \cdot (h'')^2 \geq 4h' \cdot C^* \log^6 n.$$

Notice that $C^* \geq r'h'/2$ must hold. Indeed, since $r' \leq r$ and $h' \leq h$, any solution to instance $\mathrm{GP}(G, r', h')$ of (r,h)-Graph Partitioning is also a feasible solution to instance $\mathrm{GP}(G, r, h)$. Since we have assumed that $r' = r^*$ and $h' = h^*$, there is a good solution to instance $\mathrm{GP}(G, r', h')$ of (r,h)-Graph Partitioning, and the value of any such good solution is at least $r'h'/2$. Since we have assumed that $C^* = \mathrm{OPT}_{\mathrm{GP}}(G, r, h)$, we get that $r'h'/2 \leq C^*$ must hold. We conclude that, if $h'' > h' \cdot \alpha(n) \cdot \log^8 n$, then:

$$r'' \cdot (h'')^2 \geq 2r'(h')^2 \log^6 n.$$

But we have already established above that $r'' \cdot (h'')^2 \leq r' \cdot (h')^2 \cdot \log^6 n$, a contradiction. □

We will now slightly modify the collection $\mathcal{H}'$ of subgraphs of G to obtain a feasible solution to instance $\mathrm{GP}(G,r,h)$, whose value is close to the value of $\mathcal{H}'$. First, for every cluster $H\in\mathcal{H}'$, if $|E(H)|>h'/2$, then we discard arbitrary edges from graph H, until $|E(H)|=h'/2$ holds. From Observation 6.14, if $C^*=\mathrm{OPT}_{\mathrm{GP}}(G,r,h)$, $r'=r^*$ and $h'=h^*$ hold, then the total number of edges in the graphs of $\mathcal{H}'$ decreases by at most factor $O(\alpha(n)\cdot\log^8 n)$ as the result of this transformation, and so $\sum_{H\in\mathcal{H}'}|E(H)|\geq\Omega\left(\frac{r''\cdot h''}{\alpha(n)\cdot\log^8 n}\right)\geq\Omega\left(\frac{C^*}{(\alpha(n))^2\log^{10}n}\right)$ holds. Also, if, at the end of this transformation, $\sum_{H\in\mathcal{H}'}|E(H)|>\frac{h'\cdot r'}{2}$ holds, then we discard arbitrary edges from the graphs in $\mathcal{H}'$ until $\sum_{H\in\mathcal{H}'}|E(H)|\leq\frac{h'\cdot r'}{2}$ holds. Since $h'r'\geq\frac{C^*}{4\log n}$, $\sum_{H\in\mathcal{H}'}|E(H)|\geq\Omega\left(\frac{C^*}{(\alpha(n))^2\log^{10}n}\right)$ continues to hold.

If $|\mathcal{H}'|\leq r'$, then we have obtained a valid solution to instance $\mathrm{GP}(G,r',h')$ of value $\Omega\left(\frac{C^*}{(\alpha(n))^2\log^{10}n}\right)$. Otherwise, we perform further transformations to the set $\mathcal{H}'$ of graphs as follows.

While $|\mathcal{H}'|>r'$, we let $H',H''\in\mathcal{H}'$ be a pair of graphs with smallest number of edges, breaking ties arbitrarily. We remove H' and H'' from $\mathcal{H}'$, and we add a new graph $H=H'\cup H''$ to $\mathcal{H}'$ instead. The procedure is terminated once $|\mathcal{H}'|=r'$ holds. We claim that at the end of this procedure, for every graph $H\in\mathcal{H}'$, $|E(H)|\leq h'$ holds. Indeed, assume otherwise. Consider the first time when a graph H with $|E(H)|>h'$ was added to $\mathcal{H}'$. Then $H=H'\cup H''$ must hold, where H',H'' are two graphs that belonged to $\mathcal{H}'$ prior to this iterations. Then at least one of these two graphs must contain more than $h'/2$ edges. From the choice of the graphs H',H'', and from the fact that $|\mathcal{H}'|>r'$ held at the beginning of the iteration, we get that, at the beginning of the iteration, there were at least r' graphs $\tilde{H}\in\mathcal{H}'$ with $|E(\tilde{H})|>h'/2$. But then $\sum_{\tilde{H}\in\mathcal{H}'}|E(\tilde{H})|>\frac{r'h'}{2}$ held at the beginning of the iteration. Since the total number of edges contained in the graphs of $\mathcal{H}'$ does not change over the course of the algorithm, we reach a contradiction, since we have ensured that, at the beginning of the algorithm, $\sum_{\tilde{H}\in\mathcal{H}'}|E(\tilde{H})|\leq\frac{h'\cdot r'}{2}$ held. We return the resulting collection $\mathcal{H}'$ of subgraphs of G, which is guaranteed to be a feasible solution to instance $\mathrm{GP}(G,r',h')$, of value at least $\Omega\left(\frac{C^*}{(\alpha(n))^2\log^{10}n}\right)$.

6.3.2 Case 2: $C^*<16n\alpha(n)\log^7 n$

In this case, we start by computing a maximal subgraph F of G, such that F is a forest, and maximum vertex degree in F is at most h. Such a graph F can be computed via a simple greedy algorithm. We start with graph F containing the set $V(F)=V(G)$ of vertices and no edges. We then consider the edges of G one by one. For each such edge $e\in E(G)$, if graph $F\cup\{e\}$ remains a forest with maximum vertex degree at most h, then we add e to F. Once every edge of G is processed, we obtain the final graph F. Let S be the set of all vertices of G that are adjacent to at least one edge of F, and denote $|S|=n'$.

We consider two subcases of Case 2. The first subcase, Case 2a happens if $|E(F)|\geq\frac{C^*}{64\alpha(n')\cdot\log^7 n'}$. In this case, we use the following simple observation, that will allow us to decompose the forest F to obtain a solution to instance $\mathrm{GP}(G,r',h')$ of (r,h)-Graph Partitioning, whose value is close to $|E(F)|$.

Observation 6.15. *There is an efficient algorithm, that, given a tree T with maximum vertex degree at most Δ and $|E(T)|\geq\Delta/2$, computes a collection $\mathcal{T}$ of vertex-disjoint subgraphs of T,*

such that for each subgraph $T' \in \mathcal{T}$, $\frac{\Delta}{2} \leq |E(T')| \leq \Delta$, *and* $\sum_{T' \in \mathcal{T}} |E(T')| \geq \frac{|E(T)|}{2}$.

Proof: We root the tree T in an arbitrary vertex v. Initially, we let $\mathcal{T} = \emptyset$. As long as $|E(T)| > \Delta$, we perform iterations. In every iteration, we consider an arbitrary vertex u in the current tree T, that is a non-leaf vertex, but all children of u are leaf vertices. Let T' be the subtree of T rooted at vertex u. Note that $1 \leq |E(T')| \leq \Delta - 1$. We add graph T' to $\mathcal{T}$, and we delete all vertices of T' from T. Note that, as the result of this iteration, the unique edge e connecting u to its parent-vertex in the tree T is deleted from T, and it does not belong to any graph in $\mathcal{T}$. We let $e' \in E(T)$ be an arbitrary edge (which must exist since $|E(T)| \geq 1$), and we say that e' is *responsible* for the deletion of the edge e. We then continue to the next iteration. The algorithm terminates once $|E(T)| \leq \Delta$ holds. We then add graph T to the collection $\mathcal{T}$ and terminate the algorithm. It is immediate to verify that, for every graph $T' \in \mathcal{T}$, $|E(T')| \leq \Delta$ holds. Moreover, if an edge e belonged to the original graph T, and it does not belong to $\bigcup_{T' \in \mathcal{T}} E(T')$, then some edge of $\bigcup_{T' \in \mathcal{T}} E(T')$ is designated as being responsible for deleting e. It is easy to verify that every edge $e' \in \bigcup_{T' \in \mathcal{T}} E(T')$ may be responsible for the deletion of at most one edge. Therefore, at the end of the algorithm, $\sum_{T' \in \mathcal{T}} |E(T')| \geq |E(T)|/2$ holds. □

Initially, we construct a collection $\mathcal{H}'$ of subgraphs of F as follows. For every tree T of the forest F, if $|V(T)| \leq h$, then we add T to $\mathcal{H}'$. Otherwise, we apply Observation 6.15 to tree T with parameter $\Delta = h$, and add the graphs in the resulting collection $\mathcal{T}$ to $\mathcal{H}'$. At the end of this algorithm, for every graph $H' \in \mathcal{H}'$, $|E(H')| \leq h$ holds, and $\sum_{H' \in \mathcal{H}'} |E(H')| \geq \frac{|E(F)|}{2} \geq \frac{C^*}{128\alpha(n') \cdot \log^7 n'}$. For every graph $H' \in \mathcal{H}'$, if $|E(H')| > h/2$, then we delete edges from H' until $|E(H')| \leq h/2$ holds. Clearly, after this transformation, $\sum_{H' \in \mathcal{H}'} |E(H')| \geq \frac{C^*}{256\alpha(n') \cdot \log^7 n'} \geq \Omega\left(\frac{C^*}{\alpha(n) \cdot \log^7 n}\right)$ holds. If $\sum_{H \in \mathcal{H}'} |E(H)| > hr/2$, then we discard arbitrary edges from the graphs in $\mathcal{H}'$, until $\sum_{H \in \mathcal{H}'} |E(H)| = hr/2$ holds. Since, if $C^* = \mathrm{OPT}_{\mathrm{GP}}(G, r, h)$, $C^* \leq rh$, we are still guaranteed that $\sum_{H' \in \mathcal{H}'} |E(H')| \geq \Omega\left(\frac{C^*}{\alpha(n) \cdot \log^7 n}\right)$ holds.

If $|\mathcal{H}'| \leq r$, then we have obtained a valid solution to instance $\mathrm{GP}(G, r, h)$ of value $\Omega\left(\frac{C^*}{\alpha(n) \cdot \log^7 n}\right)$. Otherwise, we proceed exactly like in Case 1 in order to transform $\mathcal{H}'$ into a valid solution to instance $\mathrm{GP}(G, r, h)$ of (r,h)-Graph Partitioning, without changing the total number of edges that lie in the graphs of $\mathcal{H}'$. While $|\mathcal{H}'| > r$, we let $H', H'' \in \mathcal{H}'$ be a pair of graphs with smallest number of edges, breaking ties arbitrarily. We remove H' and H'' from $\mathcal{H}'$, and we add a new graph $H = H' \cup H''$ to $\mathcal{H}'$ instead. The procedure is terminated once $|\mathcal{H}'| = r$ holds. We claim that at the end of this procedure, for every graph $H \in \mathcal{H}'$, $|E(H)| \leq h$ holds. Indeed, assume otherwise. Consider the first time when a graph H with $|E(H)| > h$ was added to $\mathcal{H}'$. Then $H = H' \cup H''$ must hold, where H', H'' are two graphs that belonged to $\mathcal{H}'$ prior to this iterations. Then at least one of these two graphs must contain more than $h/2$ edges. From the choice of the graphs H', H'', and from the fact that $|\mathcal{H}'| > r$ held at the beginning of the iteration, we get that, at the beginning of the iteration, there were at least r graphs $\tilde{H} \in \mathcal{H}'$ with $|E(\tilde{H})| > h/2$. But then $\sum_{\tilde{H} \in \mathcal{H}'} |E(\tilde{H})| > \frac{rh}{2}$ held at the beginning of the iteration. Since the total number of edges contained in the graphs of $\mathcal{H}'$ does not change over the course of the algorithm, we reach a contradiction, since we have ensured that, at the beginning of the algorithm, $\sum_{\tilde{H} \in \mathcal{H}'} |E(\tilde{H})| \leq \frac{h \cdot r}{2}$ held. We return the resulting set $\mathcal{H}'$ of subgraphs of G, which is guaranteed to be a feasible solution to instance $\mathrm{GP}(G, r, h)$ of (r,h)-Graph Partitioning, of value at least $\Omega\left(\frac{C^*}{\alpha(n) \log^7 n}\right)$.

It now remains to consider Case (2b), where $C^* < 16n\alpha(n)\log^7 n$ and $|E(F)| < \frac{C^*}{64\alpha(n')\cdot\log^7 n'}$. Recall that S is the set of all vertices of G that are adjacent to at least one edge of F, and recall that we have denoted $|S| = n'$.

In this case, we let $G' = G[S]$, and we consider instance $\mathrm{GP}(G', r, h)$ of (r,h)-Graph Partitioning. Notice that, if $\mathcal{H}'$ is a valid solution to instance $\mathrm{GP}(G', r, h)$ of (r,h)-Graph Partitioning, then it is also a valid solution to instance $\mathrm{GP}(G, r, h)$ of (r,h)-Graph Partitioning. We start by showing that $\mathrm{OPT}_{\mathrm{GP}}(G', r, h)$ is close to C^*.

Observation 6.16. *If $C^* = \mathrm{OPT}_{\mathrm{GP}}(G, r, h)$, and Case (2b) happens, then $\mathrm{OPT}_{\mathrm{GP}}(G', r, h) \geq \frac{C^*}{2}$.*

Proof: Let $S' \subseteq S$ be the set of all vertices whose degree in F is h, and let S^* the set of all vertices of F that are isolated.

Let $\mathcal{H}$ be the optimal solution to instance $\mathrm{GP}(G, r, h)$, and let $E' = \bigcup_{H\in\mathcal{H}} E(H)$. We partition the set E' of edges into two subsets: set E'_1 containing all edges that lie in $G' = G[S]$, and set E'_2 containing all remaining edges. Clearly, for every edge $e = (x, y) \in E'_2$, at least one endpoint of e must lie in S^*. Assume w.l.o.g. that $x \in S^*$. We claim that $y \in S'$ must hold. Indeed, otherwise $F \cup \{e\}$ remains a forest, with maximum vertex degree at most h, contradicting the fact that F is a maximal subgraph of G with these properties.

Therefore, very edge of E'_2 connects a vertex of S^* to a vertex of S'. We claim that $|E'_2| \leq |S'| \cdot h$. Indeed, from the definition of the (r,h)-Graph Partitioning problem, for every graph $H \in \mathcal{H}$, $|E(H)| \leq h$, and all graphs in $\mathcal{H}$ are disjoint in their vertices. Therefore, every vertex of G may be incident to at most h edges of E'. Since every edge of E'_2 has a vertex of S' as its endpoint, we get that $|E'_2| \leq |S'| \cdot h$.

Recall that, from our definition, every vertex $v \in S'$ has degree h in F. Therefore, $|E(F)| \geq (h-1)\cdot|S'|$. We conclude that $|E(F)| \geq |E'_2|/2$, and so $|E'_2| \leq 2|E(F)| < \frac{C^*}{4}$. Since $|E'| = C^*$, we get that $|E'_1| \geq C^*/2$.

We now define a solution $\mathcal{H}'$ to instance $\mathrm{GP}(G', h, r)$ of (r,h)-Graph Partitioning. For every graph $H \in \mathcal{H}$, we let H' be a graph that is obtained from H by deleting all vertices of S^* from it, and we let $\mathcal{H}' = \{H' \mid H \in \mathcal{H}\}$. It is easy to verify that $\mathcal{H}'$ is a valid solution to instance $\mathrm{GP}(G', r, h)$, and that its value is at least $|E' \setminus E'_2| \geq \frac{C^*}{2}$. We conclude that $\mathrm{OPT}_{\mathrm{GP}}(G', r, h) \geq \frac{C^*}{2}$. □

Denote $C' = \mathrm{OPT}_{\mathrm{GP}}(G', r, h)$. From the above discussion $\frac{C^*}{2} \leq C' \leq C^*$. Notice that for every tree T of F, if T is not a singleton vertex, then $|E(T)| \geq |V(T)| - 1 \geq \frac{|V(T)|}{2}$. Therefore, $|E(F)| \geq \frac{|S|}{2} = \frac{n'}{2}$. On the other hand, from our assumption, $|E(F)| < \frac{C^*}{64\alpha(n')\cdot\log^7 n'}$. We then conclude that $C^* > 32n'\alpha(n')\cdot\log^7 n$.

We will now focus on solving instance $\mathrm{GP}(G', r, h)$ of the (r,h)-Graph Partitioning problem. As before, we will try all guesses C^{**} on the value C' of the optimal solution for this problem. Note that we only need to consider values C^{**} that are integers, with $\frac{C^*}{2} \leq C^{**} \leq C^*$. Furthermore, from the above discussion, for each such guess, $C^{**} \geq \frac{C^*}{2} \geq 16n'\alpha(n')\cdot\log^7 n$ holds, so Case 1 will occur. We execute the algorithm from Case 1 for each such guessed value C^{**} and output the best among the resulting solutions. We are then guaranteed to obtain a solution $\mathcal{H}$ to instance $\mathrm{GP}(G', r, h)$ of value at least $\Omega\left(\frac{C'}{(\alpha(n))^2\log^{10} n}\right) \geq \Omega\left(\frac{C^*}{(\alpha(n))^2\log^{10} n}\right)$. Clearly, $\mathcal{H}$ is also a valid solution to instance $\mathrm{GP}(G, r, h)$. Overall, we obtain an efficient $O((\alpha(n))^2 \cdot \mathrm{poly}\log n)$-approximation algorithm for

(r,h)-Graph Partitioning.

7 Acknowledgement

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

A Proof of Lemma 2.1

We prove each of the directions of the reductions separately, in the following two subsections.

A.1 Reduction from Bipartite Densest (k_1, k_2)-Subgraph to Densest k-Subgraph

Assume that exists 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. We show an $O(\alpha(N^2))$-approximation algorithm for the Bipartite Densest (k_1, k_2)-Subgraph problem, whose running time is at most $O(T(N^2) \cdot \text{poly}(N))$, where N is the number of vertices in the input graph.

Let $\text{DkS}(G, k_1, k_2)$ be the input instance to the Bipartite Densest (k_1, k_2)-Subgraph problem. Denote $G = (A, B, E)$, so $|A \cup B| = N$. We construct another bipartite graph $H = (A', B', E')$, that will serve as input to the Densest k-Subgraph problem, as follows. We define, for every vertex $u \in A$, a collection $T_u = \{u^1, \ldots, u^{k_2}\}$ of vertices that we call *copies of* u, and we let $A' = \bigcup_{u \in A} T_u$. Similarly, we define, for every vertex $v \in B$, a set $T_v = \{v^1, \ldots, v^{k_1}\}$ of k_1 vertices, that we call *copies of* v, and we let $B' = \bigcup_{v \in B} T_v$. The set E' of edges of H contains, for every edge $e = (u, v) \in E(G)$, all edges in $T_u \times T_v$. Note that $|V(H)| \leq \max\{k_1, k_2\} \cdot |A \cup B| \leq N^2$.

Let $k = 2k_1k_2$, and consider the instance $\text{DkS}(H, k)$ of the Densest k-Subgraph problem. We use the following observation to lower-bound its optimal solution cost.

Observation A.1. $\text{OPT}_{\text{DkS}}(H, k) \geq k_1k_2 \cdot \text{OPT}_{\text{BDkS}}(G, k_1, k_2)$.

Proof: Let S^* be the optimal solution to instance $\text{BDkS}(G, k_1, k_2)$ of Bipartite Densest (k_1, k_2)-Subgraph. Denote $S^*_A = S^* \cap A$ and $S^*_B = S^* \cap B$, so $|S^*_A| = k_1$ and $|S^*_B| = k_2$ hold. We define $T^*_A = \bigcup_{u \in S^*_A} T_u$ and $T^*_B = \bigcup_{v \in S^*_B} T_v$. From the construction of H, it is clear that $|T^*_A| = |T^*_B| = k_1 \cdot k_2$, and $|E_H(T^*_A, T^*_B)| = k_1k_2 \cdot |E_G(S^*_A, S^*_B)|$. Therefore, $T^*_A \cup T^*_B$ is a feasible solution to instance $\text{DkS}(H, k)$ of Densest k-Subgraph, and so $\text{OPT}_{\text{DkS}}(H, 2k_1k_2) \geq k_1k_2 \cdot \text{OPT}_{\text{BDkS}}(G, k_1, k_2)$. □

In order to complete the reduction, we need the following claim.

Claim A.2. *There is an efficient algorithm, that, given any solution W to the instance* $\text{DkS}(H, k)$ *of the* Densest k-Subgraph *problem, computes a solution to instance* $\text{BDkS}(G, k_1, k_2)$ *of* Bipartite Densest (k_1, k_2)-Subgraph*, whose value is at least $|E_H(W)|/(4k_1k_2)$.*

Proof: Denote $W_A = W \cap A$ and $W_B = W \cap B$. We start by computing a partition $(W^0_A, \ldots, W^{2k_2-1}_A)$ of the vertices of W_A into $2k_2$ subsets, containing at most k_1 vertices each, so that, for every vertex u of A, no two copies of u appear in the same subset.

In order to do so, we let σ be an arbitrary ordering of the vertices of W_A, in which, for every vertex $u \in A$, all copies of u that belong to W_A appear consecutively. For all $0 \leq i \leq 2k_2 - 1$, we let $W^i_A \subseteq W_A$ to be the set of all vertices $x \in W_A$, whose index is $i \pmod{2k_2}$ in this ordering. Since, for every vertex $u \in A$, $|T_u| = k_2$, it is immediate to verify that all copies of u in W_A lie in distinct sets. It is also immediate to verify that, for all $0 \leq i < 2k_2$, $|W^i_A| \leq k_1$.

We similarly compute a partition $(W^0_B, \ldots, W^{2k_1-1}_B)$ of the vertices of W_B into $2k_1$ subsets, containing at most k_2 vertices each, so that, for every vertex v of B, no two copies of v appear in the same subset.

Let $0 \leq i^* < 2k_2, 0 \leq j^* < 2k_1$ be a pair of indices, for which $|E_H(W_A^{i^*}, W_B^{j^*})|$ is maximized. Clearly, $|E_H(W_A^{i^*}, W_B^{j^*})| \geq |E_H(W)|/(4k_1k_2)$. Finally, let $X \subseteq V(G)$ be the set of vertices containing every vertex $u \in V(G)$, whose copy lies in $W_A^{i^*} \cup W_B^{j^*}$. Note that $|X \cap A| = |W_A^{i^*}| \leq k_1$ and $|X \cap B| = |W_B^{j^*}| \leq k_2$ must hold, so X is a valid solution to instance $\mathrm{BDkS}(G, k_1, k_2)$ of Bipartite Densest (k_1, k_2)-Subgraph. Since $W_A^{i^*} \cup W_B^{j^*}$ contains at most one copy of every vertex of $V(G)$, from the construction of graph H, it is easy to verify that $|E_G(X)| = |E_H(W_A^{i^*}, W_B^{j^*})| \geq |E_H(W)|/(4k_1k_2)$. □

We are now ready to complete our reduction. We apply the approximation algorithm $\mathcal{A}$ for the Densest k-Subgraph problem to instance $\mathrm{DkS}(H, k)$, to obtain a solution W. Since $|V(H)| \leq N^2$, and since $\mathcal{A}$ is a factor-$\alpha(n)$ approximation algorithm, from Observation A.1, we get that:

$$|E_H(W)| \geq \frac{\mathrm{OPT}_{\mathrm{DkS}}(H, k)}{\alpha(N^2)} \geq \frac{k_1 \cdot k_2 \cdot \mathrm{OPT}_{\mathrm{BDkS}}(G, k_1, k_2)}{\alpha(N^2)}.$$

Additionally, the running time of the algorithm is $O(T(N^2))$.

We then apply the algorithm from Claim A.2, whose running time is bounded by $O(\mathrm{poly}(N))$ to solution W to instance $\mathrm{DkS}(H, k)$, to obtain a solution X to instance $\mathrm{DkS}(G, k_1, k_2)$ of Bipartite Densest (k_1, k_2)-Subgraph. We are guaranteed that:

$$|E_G(X)| \geq \frac{|E_H(W)|}{4k_1k_2} \geq \Omega\left(\frac{\mathrm{OPT}_{\mathrm{DkS}}(G, k_1, k_2)}{\alpha(N^2)}\right).$$

It is easy to verify that the running time of the algorithm is bounded by $O(T(N^2) \cdot \mathrm{poly}(N))$.

A.2 Reduction from Densest k-Subgraph to Bipartite Densest (k_1, k_2)-Subgraph

We now assume that there exists an efficient $\alpha(N)$-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. We show that 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.

Let $\mathrm{DkS}(G, k)$ be the input instance for the Densest k-Subgraph problem, so $|V(G)| = n$. We construct a bipartite graph $H = (V_1, V_2, E)$, where the vertex sets are $V_1 = \{u^1 \mid u \in V(G)\}$, $V_2 = \{u^2 \mid u \in V(G)\}$, and the edge set is $E = \{(u^1, v^2), (u^2, v^1) \mid (u, v) \in E(G)\}$. We denote $N = |V(H)| = 2n$. Consider the instance $\mathrm{DkS}(H, k_1, k_2)$ of Bipartite Densest (k_1, k_2)-Subgraph, where $k_1 = k_2 = k$. We use the following observation to lower-bound the optimal solution cost of this instance.

Observation A.3. $\mathrm{OPT}_{\mathrm{BDkS}}(H, k_1, k_2) \geq 2 \cdot \mathrm{OPT}_{\mathrm{DkS}}(G, k)$.

Proof: Let V^* be the optimal solution to instance $\mathrm{DkS}(G, k)$ of Densest k-Subgraph. We define $U^* = \{v^1, v^2 \mid v \in V^*\}$, so $|U^* \cap V_1| = k$ and $|U^* \cap V_2| = k$. Clearly, U^* is a feasible solution to instance $\mathrm{BDkS}(H, k, k)$. Moreover, it is easy to verify that $|E_H(U^*)| = 2 \cdot |E_G(V^*)|$, so $\mathrm{OPT}_{\mathrm{BDkS}}(H, k_1, k_2) \geq 2 \cdot \mathrm{OPT}_{\mathrm{DkS}}(G, k)$. □

We apply Algorithm $\mathcal{A}'$ to instance, $\mathrm{DkS}(H, k_1, k_2)$ of Bipartite Densest (k_1, k_2)-Subgraph, obtaining a solution U'. We denote $U_1 = U' \cap V_1$ and $U_2 = U' \cap V_2$, so $|U_1| = |U_2| = k$, and $|E_H(U_1, U_2)| \geq \mathrm{OPT}_{\mathrm{BDkS}}(H, k, k)/\alpha(N) \geq 2\mathrm{OPT}_{\mathrm{DkS}}(G, k)/\alpha(2n)$ from Observation A.3.

Let $U = \{v \mid v^1 \in U_1 \text{ or } v^2 \in U_2\}$ be a subset of vertices of G. Clearly, $|U| \leq 2k$, and $|E_G(U)| \geq \frac{|E_H(U^*)|}{2} \geq \frac{\mathrm{OPT}_{\mathrm{DkS}}(G,k)}{\alpha(2n)}$.

We then apply the algorithm from Lemma 2.3 to graph G and set U of vertices, with parameter $\beta = 1/2$, to obtain a set $\tilde{U} \subseteq U$ of vertices, with $|\tilde{U}| \leq k$, and $|E_G(\tilde{U})| \geq \Omega(|E_G(U)|) \geq \Omega(\mathrm{OPT}_{\mathrm{DkS}}(G, k)/\alpha(2n))$. Therefore, we obtained an $O(\alpha(2n))$-approximate solution to instance $\mathrm{DkS}(G, k)$ of the Densest k-Subgraph problem.

B Reduction from (r,h)-Graph Partitioning to Densest k-Subgraph

In this section we complete the proof Theorem 4.1, by showing a reduction from (r,h)-Graph Partitioning to Densest k-Subgraph. The reduction is very similar to the reduction from Dense k-Coloring to Densest k-Subgraph described in Section 4.

We start by formulating an LP-relaxation of the problem, whose number of constraints is bounded by $O(N)$, but the number of variables may be large. We then show an LP-rounding algorithm for this LP-relaxation, whose running time is $O(\mathrm{poly}(N))$ if it is given a solution to the LP-relaxation whose support size is bounded by $O(\mathrm{poly}(N))$. In order to compute an approximate LP-solution whose support size is sufficiently small, we design an approximate separation oracle for the dual of the LP-relaxation. We start with describing the LP-relaxation and providing an LP-rounding algorithm for it.

B.1 Linear Programming Relaxation and an LP-Rounding Algortihm

Let $\mathrm{GP}(G, r, h)$ be the input instance of (r,h)-Graph Partitioning, and denote $|V(G)| = N$. We let $\mathcal{H}$ be the collection of all subgraphs $H \subseteq G$ with $|E(H)| \leq h$. For each such subgraph H, we denote $m(H) = |E(H)|$. We consider the following LP-relaxation of the (r,h)-Graph Partitioning problem, that has a variable x_H for every graph $H \in \mathcal{H}$.

$$
\begin{aligned}
&\text{(LPW-P)} \\
&\max \quad \textstyle\sum_{H \in \mathcal{H}} m(H) \cdot x_H \\
&\text{s.t.} \\
&\qquad \textstyle\sum_{\substack{H \in \mathcal{H}: \\ v \in V(H)}} x_H \leq 1 \qquad \forall v \in V(G) \\
&\qquad \textstyle\sum_{H \in \mathcal{H}} x_H \leq r \\
&\qquad x_H \geq 0 \qquad\qquad \forall H \in \mathcal{H}
\end{aligned}
$$

It is easy to verify that (LPW-P) is an LP-relaxation of the (r,h)-Graph Partitioning problem. Indeed, consider a solution $(H_1, \ldots, H_r)$ to the input instance $\mathrm{GP}(G, r, h)$. For all $1 \leq i \leq r$, we set $x_{H_i} = 1$, and for every other graph $H \in \mathcal{H}$, we set $x_H = 0$. This provides a feasible solution to (LPW-P),

whose value is precisely $\sum_{i=1}^{r}|E(H)|$. We denote the value of the optimal solution to (LPW-P) by $\mathrm{OPT}_{\text{LP-P}}$. From the above discussion, $\mathrm{OPT}_{\text{LP-P}} \geq \mathrm{OPT}_{\mathrm{GP}}(G,r,h)$.

In the following claim we provide an LP-rounding algorithm for (LPW-P). The claim is an analogue of Claim 4.2. Its proof is almost identical and is provided here for completeness.

Claim B.1. *There is an efficient randomized algorithm, whose input consists of an instance* $\mathrm{GP}(G,r,h)$ *of the* (r,h)-Graph Partitioning *problem with* $N=|V(G)|$*, such that* N *is greater than a large enough constant, and a solution* $\{x_H \mid H\in\mathcal{H}\}$ *to (LPW-P), in which the number of variables* x_H *with* $x_H>0$ *is bounded by* $O(\mathrm{poly}(N))$*, and* $\sum_{H\in\mathcal{H}} m(H)\cdot x_H \geq OPT_{\text{LP-P}}/\beta$*, for some parameter* $1\leq\beta\leq N^3$*; the solution is given by only specifying values of variables* x_H *that are non-zero. The algorithm with high probability returns an integral solution* $(H_1,\ldots,H_r)$ *to instance* $\mathrm{GP}(G,r,h)$*, such that* $\sum_{i=1}^{r}|E(H_i)| \geq \frac{\mathrm{OPT}_{\mathrm{GP}}(G,r,h)}{2000\beta\log^3 N}$.

Proof: We assume that we are given a solution $\{x_H \mid H\in\mathcal{H}\}$ to (LPW-P), in which the number of variables x_H with $x_H>0$ is bounded by $O(\mathrm{poly}(N))$. Denote $C=\sum_{H\in\mathcal{H}} m(H)\cdot x_H$, and recall that $C\geq \mathrm{OPT}_{\text{LP-P}}/\beta$ holds. We denote by $\mathcal{H}'\subseteq\mathcal{H}$ the collection of all graphs $H\in\mathcal{H}$ with $x_H>0$.

We construct another collection $\mathcal{H}''\subseteq\mathcal{H}'$ of subgraphs of G as follows. For every subgraph $H\in\mathcal{H}'$, we add H to $\mathcal{H}''$ independently, with probability x_H. Clearly, $\mathbf{E}\left[\sum_{H\in\mathcal{H}''} m(H)\right]=\sum_{H\in\mathcal{H}} m(H)\cdot x_H = C$.

We say that a bad event $\mathcal{E}_1$ happens if some vertex $v\in V(G)$ lies in more than $5\log N$ graphs in $\mathcal{H}''$. We say that a bad event $\mathcal{E}_2$ happens if $|\mathcal{H}''|>5r\log N$. We say that a bad event $\mathcal{E}_3$ happens if $\sum_{H\in\mathcal{H}''} m(H) < \frac{C}{8}$. Lastly, we say that a bad event $\mathcal{E}$ happens if either of the events $\mathcal{E}_1,\mathcal{E}_2$, or $\mathcal{E}_3$ happen. The following observation is an analogue of Observation 4.3. Its proof is identical and is omitted here.

Observation B.2. $\mathbf{Pr}\left[\mathcal{E}\right]\leq 2/N^3$.

Observe that we can efficiently check whether Event $\mathcal{E}$ happened. If Event $\mathcal{E}$ happens, then we terminate the algorithm with a FAIL. We assume from now on that Event $\mathcal{E}$ did not happen. Then $\sum_{H\in\mathcal{H}''} m(H)\geq\frac{C}{8}\geq\frac{\mathrm{OPT}_{\mathrm{GP}}(G,r,h)}{8\beta}$ must hold. We denote $\mathcal{H}''=\{H_1,H_2,\ldots,H_z\}$, where the graphs are indexed according to their value $m(H)$, so that $m(H_1)\geq m(H_2)\geq\cdots\geq m(H_z)$. We then let $\mathcal{H}^*=\{H_1,\ldots,H_r\}$ (if $z<r$, then we set $H_{z+1}=\cdots=H_r=\emptyset$). For all $1\leq i\leq r$, we denote $E_i=E(H_i)$, so $|E_i|=m(H_i)$. Recall that, since Event $\mathcal{E}$ did not happen, $|\mathcal{H}''|\leq 5r\log N$ holds. Therefore:

$$\sum_{i=1}^{r}|E_i|\geq\frac{\sum_{H\in\mathcal{H}''}m(H)}{5\log N}\geq\frac{\mathrm{OPT}_{\mathrm{GP}}(G,r,h)}{40\beta\log N}.$$

As before, the graphs in set $\mathcal{H}$ may not be mutually disjoint. However, since Event $\mathcal{E}$ did not happen, every vertex of $V(G)$ may lie in at most $5\log N$ such graphs. We now construct a new collection $\mathcal{H}^{**}=\{H_1',\ldots,H_r'\}$ of graphs, as follows. For all $1\leq i\leq r$, we will define a subset $V_i\subseteq V(H_i)$ of vertices, and we will then set $H_i'=H_i[V_i]$. In order to define vertex sets $V_1,\ldots,V_r$, we start by setting $V_1=V_2=\cdots=V_r=\emptyset$, and then process vertices $v\in V(H)$ one by one. Consider any vertex $v\in V(G)$, and let $H_{i_1},H_{i_2},\ldots,H_{i_a}\in\mathcal{H}^*$ be the graphs of $\mathcal{H}^*$ containing v.

Vertex v chooses an index $i^* \in \{i_1, \ldots, i_a\}$ at random, and is then added to V_{i^*}. Once all vertices of $V(G)$ are processed, we obtain a final collection $V_1, \ldots, V_r$ of sets of vertices, where for all $1 \leq i \leq r$, $V_i \subseteq V[H_i]$. For all $1 \leq i \leq r$, we then set $H'_i = H_i[V_i]$. Since $H'_i \subseteq H_i$, we are then guaranteed that $|E(H'_i)| \leq h$ holds.

Note that for all $1 \leq j \leq r$, for every vertex $v \in V(H_j)$, the probability that $v \in V_j$ is at least $1/(5 \log N)$. We say that an edge $e = (u, v) \in E_j$ *survives* if both $u, v \in V_j$. We denote by $E'' \subseteq \bigcup_{i=1}^{r} E_i$ the set of all edges that survive. Since $\mathbf{Pr}[u \in V_j] \geq 1/(5 \log N)$, $\mathbf{Pr}[v \in V_j] \geq 1/(5 \log N)$, and the two events are independent, we get that the probability that edge e survives is at least $1/(25 \log^2 N)$. Overall, we get that:

$$\mathbf{E}\left[|E''|\right] \geq \frac{\sum_{i=1}^{r} |E_i|}{25 \log^2 N} \geq \frac{\mathrm{OPT}_{\mathrm{GP}}(G, r, h)}{1000 \beta \log^3 N}.$$

The final solution to instance $\mathrm{GP}(G, r, h)$ is $\mathcal{H}^{**} = \{H'_1, \ldots, H'_r\}$. Clearly, the value of this solution is $|E''|$.

So far we have obtained a randomized algorithm that either returns FAIL (with probability at most $2/N^3$), or it returns a solution to instance instance $\mathrm{GP}(G, r, h)$ of the (r,h)-Graph Partitioning problem, whose expected value is at least $\frac{\mathrm{OPT}_{\mathrm{GP}}(G,r,h)}{1000 \beta \log^3 N}$.

Let p' be the probability that the algorithm returned a solution of value at least $\frac{\mathrm{OPT}_{\mathrm{GP}}(G,r,h)}{2000 \beta \log^3 N}$, given that it did not return FAIL. Note that the expected solution value, assuming the algorithm did not return FAIL, is at most $\frac{\mathrm{OPT}_{\mathrm{GP}}(G,r,h)}{2000 \beta \log^3 N} + p' \cdot \mathrm{OPT}_{\mathrm{GP}}(G, r, h)$. Since this expectation is also at least $\frac{\mathrm{OPT}_{\mathrm{GP}}(G,r,h)}{1000 \beta \log^3 N}$, we get that $p' \geq \frac{1}{1000 \beta \log^3 N}$. Overall, the probability that our algorithm successfully returns a solution of value at least $\frac{\mathrm{OPT}_{\mathrm{GP}}(G,r,h)}{2000 \beta \log^3 N}$ is $p' \cdot \mathbf{Pr}[\neg \mathcal{E}] \geq \Omega\left(\frac{1}{\beta \log^3 N}\right)$. By repeating the algorithm $\mathrm{poly}(N)$ times we can ensure that it successfully computes a solution of value at least $\frac{\mathrm{OPT}_{\mathrm{GP}}(G,r,h)}{2000 \beta \log^3 N}$ with high probability. □

B.2 Approximately Solving the LP-Relaxation

In this subsection we provide an approximate separation oracle for the dual linear program of (LPW-P). This is sufficient in order to obtain an algorithm with running time $O(\mathrm{poly}(N))$ that approximately solves (LPW-P) using the methods described in Section 4.2. The following Linear Program is a Dual of (LPW-P). It has a variable y_v for every vertex $v \in V(G)$, and an additional variable z.

$$\begin{array}{lll}
\text{(LPW-D)} & & \\
\min & r \cdot z + \sum_{v \in V(G)} y_v & \\
\text{s.t.} & & \\
& z + \sum_{v \in V(H)} y_v \geq m(H) & \forall H \in \mathcal{H} \\
& z \geq 0 & \\
& y_v \geq 0 & \forall v \in V(G)
\end{array}$$

We denote the value of the optimal solution to (LPW-D) by $\mathrm{OPT}_{\text{LPW-D}}$.

The following lemma provides a randomized separation oracle for (LPW-D). It is an analogue of Lemma 4.4, and its proof is essentially identical. We provide it here for completeness.

Lemma B.3. *Assume that there is an efficient $\alpha(n)$-approximation algorithm for the* Densest k-Subgraph *problem, where α is an increasing function, and n is the number of vertices in the input graph. Then there is a randomized $\beta(N)$-approximate separation oracle for (LPW-D), where N is the number of variables in the input graph G, and $\beta(N) = O(\alpha(N^2) \cdot \log^2 N)$.*

Proof: Recall that we are given as input real values z and $\{y_v \mid v \in V(G)\}$. As before, we can efficiently check whether $z \geq 0$, and whether $y_v \geq 0$ for all $v \in V(G)$. If this is not the case, we can return the corresponding violated constraint.

We say that a subgraph $H \in \mathcal{H}$ is *bad* if $z + \sum_{v \in V(H)} y_v < m(H)/\beta$ holds, where $\beta = c \cdot \alpha(N^2) \cdot \log^2 N$, and c is a large enough constant whose value we set later. Our goal is to design an efficient algorithm that either returns a violated constraint of the LP (that is, a graph $H \in \mathcal{H}$ for which $z + \sum_{v \in V(H)} y_v < m(H)$ holds); or it returns "accept". We require that, if there exists a bad subgraph $H \in \mathcal{H}$, then the probability that the algorithm returns "accept" is at most $2/3$.

We slightly modify the input values in $\{y_v \mid v \in V(G)\}$, almost exactly like in the proof of Lemma B.3. First, for every vertex $v \in V(G)$ with $y_v > h$, we let y'_v be the smallest integral power of 2 that is greater than h, and for every vertex $v \in V(G)$ with $y_v < 1/4$, we set $y'_v = 0$. For each remaining vertex v, we let y'_v be the smallest integral power of 2 that is greater than $4y_v$. Notice that, for every vertex v with $y'_v \neq 0$, $1 \leq y'_v \leq 4h$ holds, and y'_v is an integral power of 2. We also set $z' = 2z$. We say that a subgraph $H \in \mathcal{H}$ is *problematic* if $z' + \sum_{v \in V(H)} y'_v < 8m(H)/\beta$ holds. We use the following two observations, that are analogues of Observation 4.5 and Observation 4.6; their proofs are also almost identical.

Observation B.4. *If $H \in \mathcal{H}$ is a bad subgraph of G, then it is a problematic subgraph of G.*

Proof: Recall that, if H is a bad subgraph, then $z + \sum_{v \in V(H)} y_v < m(H)/\beta$ must hold. Since, for every vertex $v \in V(G)$, $y'_v \leq 8y_v$, and $z' = 2z$, we get that:

$$z' + \sum_{v \in V(H)} y'_v \leq 2z + 8 \sum_{v \in V(H)} y_v \leq 8 \left(z + \sum_{v \in V(H)} y_v \right) < 8m(H)/\beta.$$

Therefore, subgraph H is problematic. □

Observation B.5. *Assume that there exists a subgraph $H \in \mathcal{H}$, for which $z' + \sum_{v \in V(H)} y'_v < m(H)$ holds. Let $H' \subseteq H$ be the graph obtained from H after removing all isolated vertices from it. Then $z + \sum_{v \in V(H')} y_v < m(H')$ holds.*

Proof: Since every vertex $v \in V(H) \setminus V(H')$ is isolated in H, we get that $m(H') = |E(H')| = |E(H)| = m(H)$. We partition the vertices of H' into two subsets: set X containing all vertices $v \in V(H')$ with $y_v < 1/4$, and set Y containing all remaining vertices. Clearly, $\sum_{v \in X} y_v < \frac{|X|}{4} \leq \frac{m(H')}{2}$ (since $m(H') \geq |V(H')|/2 \geq |X|/2$, as graph H' contains no isolated vertices).

Assume for contradiction that $z + \sum_{v \in V(H')} y_v \geq m(H')$. Then:

$$z + \sum_{v \in Y} y_v \geq m(H') - \sum_{v \in X} y_v \geq m(H')/2.$$

We now consider two cases. The first case is when there is some vertex $v \in Y$ with $y_v \geq h$. In this case, $y'_v \geq h$ holds, and $z' + \sum_{v \in S} y'_v \geq h > m(H')$ holds, a contradiction.

Otherwise, for every vertex $v \in Y$, $y'_v \geq 4y_v$ holds. Since $z' = 2z$ also holds, we get that:

$$z' + \sum_{v \in V(H)} y'_v \geq z' + \sum_{v \in Y} y'_v \geq 2z + 4 \sum_{v \in Y} y_v \geq m(H') = m(H),$$

a contradiction. □

From now on we focus on values $z', \{y'_v \mid v \in V(G)\}$. It is now enough to design an efficient randomized algorithm, that either computes a subgraph $H \in \mathcal{H}$, for which $z' + \sum_{v \in V(H)} y'_v < m(H)$ holds, or returns "accept". It is enough to ensure that, if there is a problematic subraph $H \in \mathcal{H}$, then the algorithm returns "accept" with probability at most $2/3$. Indeed, if there is a bad subgraph $H \in \mathcal{H}$, then, from Observation B.4, there is a problematic subgraph, and the algorithm will return "accept" with probability at most $2/3$. On the other hand, if the algorithm computes a subgraph $H \in \mathcal{H}$ of vertices, for which $z' + \sum_{v \in S} y'_v < m(S)$ holds, then we can return the subgraph $H' \subseteq H$ from the statement of Observation B.5, that defines a violated constraint with respect to the original LP-values.

Our algorithm is essentially the same as before: it computes a random partition (A, B) of the vertices of G, where every vertex $v \in V(G)$ is independently added to A or to B with probability $1/2$ each. Let $q = \lceil \log(8h) \rceil$. For all $1 \leq i \leq q$, we define a set $A_i \subseteq A$ of vertices: $A_i = \{v \in A \mid y'_v = 2^{i-1}\}$, and we let $A_0 = \{v \in A \mid y'_v = 0\}$. Clearly, $(A_0, \ldots, A_q)$ is a partition of the set A of vertices.

We compute a partition $(B_0, \ldots, B_q)$ of the vertices of B similarly. For all $0 \leq i, j \leq q$, we denote by $E_{i,j}$ the set of all edges $e = (u, v)$ with $u \in A_i$ and $v \in B_j$, and we define a bipartite graph $G_{i,j}$, whose vertex set is $A_i \cup B_j$, and edge set is $E_{i,j}$.

Recall that we have assumed 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. From Lemma 2.1, there exists an efficient $O(\alpha(\hat{n}^2))$-approximation algorithm for the Bipartite Densest (k_1, k_2)-Subgraph problem, where $\hat{n}$ is the number of vertices in the input graph. We denote this algorithm by $\mathcal{A}'$.

For every pair $0 \leq i, j \leq q$ of integers, and every pair $0 \leq k_1, k_2 \leq N$ of integers, we apply Algorithm $\mathcal{A}'$ for the Bipartite Densest (k_1, k_2)-Subgraph problem to graph $G_{i,j}$, with parameters k_1 and k_2. Let $S_{i,j}^{k_1,k_2}$ be the output of this algorithm, and let $m_{i,j}^{k_1,k_2}$ be the number of edges in the subgraph of $G_{i,j}$ that is induced by the set $S_{i,j}^{k_1,k_2}$ of vertices. We say that the application of algorithm $\mathcal{A}'$ is *successful* if $z' + \sum_{v \in S_{i,j}^{k_1,k_2}} y'_v < \min\{h, m_{i,j}^{k_1,k_2}\}$, and otherwise it is *unsuccessful*. If, for any quadruple (i, j, k_1, k_2) of indices, the application of algorithm $\mathcal{A}'$ was successful, then we return a graph H, that is defined as the subgraph of G induced by the set $S = S_{i,j}^{k_1,k_2}$ of vertices; if this graph contains more than h edges, then we delete arbitrary edges from it, until $|E(H)| = h$ holds.

Clearly, $H \in \mathcal{H}$ must hold. Moreover, we are guaranteed that $z' + \sum_{v \in V(H)} y'_v < \min\{h, m_{i,j}^{k_1,k_2}\} \leq m(H)$, as required. If every application of algorithm $\mathcal{A}'$ is unsuccessful, then we return "accept". The following observation will finish the proof of Lemma B.3. The observation is an analogue of Observation 4.7 and its proof is essentially identical.

Observation B.6. *Suppose there is a problematic subgraph $H \in \mathcal{H}$. Then the probability that the algorithm returns "accept" is at most* $2/3$.

Proof: Let $H \in \mathcal{H}$ be a problematic subgraph, and denote $S = V(H)$, so $z' + \sum_{v \in S} y'_v < 8m(S)/\beta$ holds. Let $E' = E(H)$, so $|E'| = m(H) \leq h$.

Denote $A_S = A \cap S, B_S = B \cap S$, and let $E'' \subseteq E'$ be the set of edges e, such that exactly one endpoint of e lies in A. Clearly, for every edge $e \in E'$, $\mathbf{Pr}\left[e \in E''\right] = 1/2$. Therefore, $\mathbf{E}\left[|E''|\right] = |E'|/2$. Let $\mathcal{E}'$ be the bad event that $|E''| < |E'|/8$. Using the same arguments as in the proof of Observation 4.7, $\mathbf{Pr}\left[\mathcal{E}'\right] \leq 2/3$. Next, we show that, if Event $\mathcal{E}'$ does not happen, then the algorithm does not return "accept".

From now on we assume that Event $\mathcal{E}'$ did not happen, so $|E''| \geq |E'|/8$. Therefore:

$$z' + \sum_{v \in S} y'_v < \frac{8m(H)}{\beta} \leq \frac{64|E''|}{\beta}$$

holds.

Clearly, there must be a pair $0 \leq i, j \leq q$ of indices, such that $|E'' \cap E_{i,j}| \geq \frac{|E''|}{4q^2} \geq \frac{|E''|}{128 \log^2 m}$. We now fix this pair i, j of indices, and denote $A'_i = A_i \cap S$ and $B'_j = B_j \cap S$. We also denote $k_1 = |A'_i|$ and let $k_2 = |B'_j|$. Denote $M_{i,j} = |E'' \cap E_{i,j}|$. Fom our choice of indices i, j, we get that:

$$z' + \sum_{v \in A'_i \cup B'_j} y'_v \leq z' + \sum_{v \in S} y'_v \leq \frac{64|E''|}{\beta} \leq \frac{M_{i,j}}{\beta} \cdot (2^{13} \cdot \log^2 m).$$

Notice that the set $S' = A'_i \cup B'_j$ of vertices provides a solution to the instance of the Bipartite Densest (k_1, k_2)-Subgraph problem on graph $G_{i,j}$ with parameters k_1, k_2, whose value is at least $M_{i,j}$. Let $S'' = S_{i,j}^{k_1,k_2}$ be the set of vertices obtained by applying Algorithm $\mathcal{A}'$ to graph $G_{i,j}$ with parameters k_1, k_2. Since $|V(G_{i,j})| \leq N$, and since $\mathcal{A}'$ is an $O(\alpha(N^2))$-approximation algorithm for Bipartite Densest (k_1, k_2)-Subgraph, we are guaranteed that $|E_G(S'')| \geq \Omega\left(\frac{M_{i,j}}{\alpha(N^2)}\right)$. Recall that $|A \cap S''| \leq k_1$; $A \cap S'' \subseteq A_i$, and all vertices $v \in A_i$ have an identical value y'_v. Therefore, $\sum_{v \in A \cap S''} y'_v \leq \sum_{v \in A'_i} y'_v$. Using a similar reasoning, $\sum_{v \in B \cap S''} y'_v \leq \sum_{v \in B'_j} y'_v$. Overall, we then get that:

$$\begin{aligned} z' + \sum_{v \in S''} y'_v &\leq z' + \sum_{v \in S'} y'_v \\ &\leq \frac{M_{i,j}}{\beta} \cdot (2^{13} \cdot \log^2 m) \\ &\leq O\left(\frac{\alpha(N^2) \cdot 2^{13} \cdot \log^2 m}{\beta}\right) \cdot \min\{|E_G(S'')|, h\}. \end{aligned}$$

Recall that $\beta = c \cdot \alpha(N^2) \cdot \log^2 N$. By letting the value of the constant c be large enough, we can ensure that $z' + \sum_{v \in S''} y'_v < \min\{|E_G(S'')|, h\}$, and so the application of algorithm $\mathcal{A}'$ to graph $G_{i,j}$ with parameters k_1 and k_2 is guaranteed to be successful. Therefore, if Event $\mathcal{E}'$ does not happen, and we set c to be a large enough constant, then our algorithm does not return "accept". Since $\mathbf{Pr}[\mathcal{E}'] \leq 2/3$, the observation follows. □ □

We can use the separation oracle described in Lemma B.3 in order to obtain a $\beta(N)$-approximate solution to (LPW-P), whose support size is bounded by $O(\text{poly}(N))$ using the standard techniques that were described in Section 4.2; we do not repeat them here. By applying the LP-rounding algorithm from Claim B.1 to the resulting LP-solution, with high probability we obtain, in time $O(\text{poly}(N))$, an integral solution $(H_1, \ldots, H_r)$ to instance $\text{GP}(G, r, h)$, such that $\sum_{i=1}^{r} |E(H_i)| \geq \Omega\left(\frac{\text{OPT}_{\text{GP}}(G,r,h)}{300\beta(N)\log^3 N}\right)$. Since $\beta(N) = O(\alpha(N^2) \cdot \log^2 N)$, with high probability we obtain an $O(\alpha(N^2) \cdot \text{poly}\log N)$-approximate solution to the input instance of (r,h)-Graph Partitioning.

References

[AAM+11] 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.

[ABS15] 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.

[ACNS82] M. Ajtai, V. Chvátal, M. Newborn, and E. Szemerédi. Crossing-free subgraphs. *Theory and Practice of Combinatorics*, pages 9–12, 1982.

[ADD+93] Ingo Althöfer, Gautam Das, David Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9(1):81–100, 1993.

[AMS07] 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.

[ARV09] Sanjeev Arora, Satish Rao, and Umesh V. Vazirani. Expander flows, geometric embeddings and graph partitioning. *J. ACM*, 56(2), 2009.

[Bar15] 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.

[BCC+10] 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.

[BCG+12] 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.

[Ber45] Joseph Bertrand. *Memory on the number of values a function can take: when you swap the letters it contains.* Bachelor, 1845.

[BGH+15] 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.

[BKRW17] 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.

[BKS19] 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.

[Bol04] Béla Bollobás. *Extremal graph theory.* Courier Corporation, 2004.

[Cab13] Sergio Cabello. Hardness of approximation for crossing number. *Discrete & Computational Geometry*, 49(2):348–358, 2013.

[CCH+20] 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.

[CDK+18] 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.

[Čeb50] Pafnutij Lvovič Čebyšev. Mémoire sur les nombres premiers. 1850.

[CH11] 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.

[Chu11] 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.

[Chu15] 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.

[CKN21] 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.

[CMS11] 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.

[CMT20] 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.

[CS13] 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.

[CT22] 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.

[DKK+18a] 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.

[DKK+18b] 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.

[DP09] Devdatt P Dubhashi and Alessandro Panconesi. *Concentration of measure for the analysis of randomized algorithms*. Cambridge University Press, 2009.

[EGS02] 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.

[Fei02] 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.

[FL01] Uriel Feige and Michael Langberg. Approximation algorithms for maximization problems arising in graph partitioning. *Journal of Algorithms*, 41(2):174–211, 2001.

[FPK01] Uriel Feige, David Peleg, and Guy Kortsarz. The dense k-subgraph problem. *Algorithmica*, 29(3):410–421, 2001.

[FS+97] 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`.

[GL09] Doron Goldstein and Michael Langberg. The dense k subgraph problem. *arXiv preprint arXiv:0912.5327*, 2009.

[Han22] Tesshu Hanaka. Computing densest k-subgraph with structural parameters. *arXiv preprint arXiv:2207.09803*, 2022.

[Hås01] Johan Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4):798–859, 2001.

[Kho02] 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.

[Kho06] Subhash Khot. Ruling out ptas for graph min-bisection, dense k-subgraph, and bipartite clique. *SIAM Journal on Computing*, 36(4):1025–1071, 2006.

[KLS00] Sanjeev Khanna, Nathan Linial, and Shmuel Safra. On the hardness of approximating the chromatic number. *Combinatorica*, 20(3):393–415, 2000.

[KMMS18] 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`.

[KMS17] 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.

[KMS18] 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.

[KP93] 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.

[KS13] Subhash Khot and Muli Safra. A two-prover one-round game with strong soundness. *Theory of Computing*, 9:863–887, 2013.

[KS17] 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.

[KS19] 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.

[Lei83] F. T. Leighton. *Complexity issues in VLSI: optimal layouts for the shuffle-exchange graph and other networks*. MIT Press, 1983.

[Lin18] Bingkai Lin. The parameterized complexity of the k-biclique problem. *Journal of the ACM (JACM)*, 65(5):1–23, 2018.

[LR99] 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.

[LT79] Richard J Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.

[Man17] 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.

[Man18] 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.

[Mat02] J. Matoušek. *Lectures on discrete geometry*. Springer-Verlag, 2002.

[PSS96] János Pach, Farhad Shahrokhi, and Mario Szegedy. Applications of the crossing number. *Algorithmica*, 16(1):111–117, 1996.

[PT00] J. Pach and G. Tóth. Thirteen problems on crossing numbers. *Geombinatorics*, 9(4):194–207, 2000.

[RS09] 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.

[Sch12] Marcus Schaefer. The graph crossing number and its variants: A survey. *The electronic journal of combinatorics*, pages DS21–Sep, 2012.

[Sot20] Renata Sotirov. On solving the densest k-subgraph problem on large graphs. *Optimization Methods and Software*, 35(6):1160–1178, 2020.

[Ste] 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.

[Tur77] P. Turán. A note of welcome. *J. Graph Theory*, 1:1–5, 1977.