From Gap-ETH to FPT-Inapproximability: Clique, Dominating Set, and More

Parinya Chalermsook* Marek Cygan† Guy Kortsarz‡ Bundit Laekhanukit§
Pasin Manurangsi¶ Danupon Nanongkai‖ Luca Trevisan**

January 4, 2018

Abstract

We consider questions that arise from the intersection between the areas of polynomial-time approximation algorithms, subexponential-time algorithms, and fixed-parameter tractable algorithms. The questions, which have been asked several times (e.g., [Mar08; FGMS12; DF13]) are whether there is a non-trivial *FPT-approximation* algorithm for the Maximum Clique (Clique) and Minimum Dominating Set (DomSet) problems parameterized by the size of the optimal solution. In particular, letting OPT be the optimum and N be the size of the input, is there an algorithm that runs in $t(\mathsf{OPT})\,\mathrm{poly}(N)$ time and outputs a solution of size $f(\mathsf{OPT})$, for any functions t and f that are independent of N (for Clique, we want $f(\mathsf{OPT}) = \omega(1)$)?

In this paper, we show that both Clique and DomSet admit no non-trivial FPT-approximation algorithm, i.e., there is no $o(\mathsf{OPT})$-FPT-approximation algorithm for Clique and no $f(\mathsf{OPT})$-FPT-approximation algorithm for DomSet, for any function f (e.g., this holds even if f is an exponential or the Ackermann function). In fact, our results imply something even stronger: The best way to solve Clique and DomSet, even approximately, is to essentially enumerate all possibilities. Our results hold under the *Gap Exponential Time Hypothesis* (Gap-ETH) [Din16; MR16], which states that no $2^{o(n)}$-time algorithm can distinguish between a satisfiable 3SAT formula and one which is not even $(1-\varepsilon)$-satisfiable for some constant $\varepsilon > 0$.

Besides Clique and DomSet, we also rule out non-trivial FPT-approximation for Maximum Biclique, the problem of finding maximum subgraphs with hereditary properties (e.g., Maximum Induced Planar Subgraph), and Maximum Induced Matching in bipartite graphs. Previously only exact versions of these problems were known to be W[1]-hard [Lin15; KR00; MS09]. Additionally, we rule out $k^{o(1)}$-FPT-approximation algorithm for Densest k-Subgraph although this ratio does not yet match the trivial $O(k)$-approximation algorithm.

To the best of our knowledge, prior results only rule out constant factor approximation for Clique [HKK13; BEKP15] and $\log^{1/4+\epsilon}(\mathsf{OPT})$ approximation for DomSet for any constant $\epsilon > 0$ [CL16]. Our result on Clique significantly improves on [HKK13; BEKP15]. However, our result on DomSet is incomparable to [CL16] since their results hold under ETH while our results hold under Gap-ETH, which is a stronger assumption.

*Aalto University, Finland. Email: parinya.chalermsook@aalto.fi.
†Institute of Informatics, University of Warsaw, Poland. Email: cygan@mimuw.edu.pl.
‡Rutgers University-Camden, New Jersey, USA. Email:guyk@camden.rutgers.edu.
§Shanghai University of Finance and Economics. Email: bundit.laekhanukit@mail.mcgill.ca
¶University of California, Berkeley, USA. Email: pasin@berkeley.edu.
‖KTH Royal Institute of Technology, Sweden. Email: danupon@kth.se
**University of California, Berkeley, USA. Email: luca@berkeley.edu.
[0]The preliminary version of this paper appears in FOCS 2017 [CCKLMNT17].

Contents

1 Introduction

Fixed-parameter approximation algorithm (in short, FPT-approximation algorithm) is a new concept emerging from a cross-fertilization between two trends in coping with NP-hard problems: *approximation algorithms* and *fixed-parameter tractable (FPT) algorithms*. Roughly speaking, an FPT-approximation algorithm is similar to an FPT algorithm in that its running time can be of the form $t(\mathsf{OPT})\,\mathrm{poly}(N)$ time (called the *FPT time*), where t is any function (possibly super exponentially growing), N is the input size, and OPT is the value of the optimal solution[1]. It is similar to an approximation algorithm in that its output is an *approximation* of the optimal solution; however, the approximation factor is analyzed in terms of the optimal solution (OPT) and *not* the input size (N). Thus, an algorithm for a maximization (respectively, minimization) problem is said to be $f(\mathsf{OPT})$-*FPT-approximation* for some function f if it outputs a solution of size at least $\mathsf{OPT}/f(\mathsf{OPT})$ (respectively, at most $\mathsf{OPT}\cdot f(\mathsf{OPT})$). For a maximization problem, such an algorithm is *non-trivial* if $f(\mathsf{OPT})$ is $o(\mathsf{OPT})$ while for a minimization problem, it is non-trivial if f is a computable function depending only on OPT.

The notion of FPT-approximation is useful when we are interested in a small optimal solution, and in particular its existence connects to a fundamental question *whether there is a non-trivial approximation algorithm when the optimal solution is small*. Consider, for example, the *Maximum Clique* (Clique) problem, where the goal is to find a clique (complete subgraph) with maximum number of vertices in an n-vertex graph G. By outputting any single vertex, we get a trivial polynomial-time n-approximation algorithm. The bound can be improved to $O(\frac{n}{\log n})$ and even to $O(\frac{n(\log\log n)^2}{\log^3 n})$ with clever ideas [Fei04]. Observe, however, that these bounds are quite meaningless when $\mathsf{OPT} = O(\frac{n(\log\log n)^2}{\log^3 n})$ since outputting a single vertex already guarantees such bounds. In this case, a bound such as $O(\frac{\mathsf{OPT}}{\log\log \mathsf{OPT}})$ would be more meaningful. Unfortunately, no approximation ratio of the form $o(\mathsf{OPT})$ is known even when FPT-time is allowed[2] (Note that outputting a single vertex gives an OPT-approximation guarantee.)

Similar questions can be asked for a minimization problem. Consider for instance, *Minimum Dominating Set* (DomSet): Find the smallest set of vertices S such that every vertex in an n-vertex input graph G has a neighbor in S. DomSet admits an $O(\log n)$-approximation algorithm via a basic greedy method. However, if we want the approximation ratio to depend on OPT and not n, no $f(\mathsf{OPT})$-approximation ratio is known for any function f (not even $2^{2^{\mathsf{OPT}}}$).

In fact, the existence of non-trivial FPT-approximation algorithms for Clique and DomSet has been raised several times in the literature (e.g., [Mar08; FGMS12; DF13]). So far, the progress towards these questions can only rule out $O(1)$-FPT-approximation algorithms for Clique. This was shown independently by Hajiaghayi et al. [HKK13] and Bonnet et al. [BEKP15], assuming the *Exponential Time Hypothesis* (ETH), which asserts that no subexponential time algorithms can decide whether a given 3SAT formula is satisfiable, and that a *linear-size PCP* exists. Alternatively, Khot and Shinkar [KS16] proved this under a rather non-standard assumption that solving quadratic equations over a finite field under a certain regime of parameters is not in FPT; unfortunately, this

[1] There are many ways to parameterize a problem. In this paper we focus on the *standard parameterization* which parameterizes the optimal solution.

[2] In fact, for maximization problems, it can be shown that a problem admits an $f(\mathsf{OPT})$-FPT-approximation algorithm for some function $f = o(\mathsf{OPT})$ if and only if it admits a polynomial-time algorithm with approximation ratio $f'(\mathsf{OPT})$ for some function $f' = o(\mathsf{OPT})$ [GG07; Mar08] (also see [Mar13]). So, it does not matter whether the running time is polynomial on the size of the input or depends on OPT.

assumption was later shown to be false [Kay14]. For DomSet, Chen and Li [CL16] could rule out $O(1)$-FPT-approximation algorithms assuming FPT $\neq$ W[1]. Moreover, they improved the inapproximability ratio to $\log^{1/4+\epsilon}(\mathsf{OPT})$ for any constant $\epsilon > 0$ under ETH. Remark that ETH implies FPT $\neq$ W[1].

Our Results and Techniques. We show that there is no non-trivial FPT-approximation algorithm for both Clique and DomSet. That is, there is no $o(\mathsf{OPT})$-FPT-approximation algorithm for Clique and no $f(\mathsf{OPT})$-FPT-approximation algorithm for DomSet, for any computable function f. Our results hold under the *Gap Exponential Time Hypothesis* (Gap-ETH), which states that distinguishing between a satisfiable 3SAT formula and one which is not even $(1-\epsilon)$-satisfiable requires exponential time for some constant $\epsilon > 0$ (see Section 2).

Gap-ETH, first formalized in [Din16; MR16], is a stronger version of the aforementioned ETH. It has recently been shown to be useful in proving fine-grained hardness of approximation for problems such as dense CSP with large alphabets [MR16] and Densest-k-Subgraph with perfect completeness [Man17a].

Note that Gap-ETH is implied by ETH if we additionally assume that a linear-size PCP exists. So, our result for Clique significantly improves the results in [HKK13; BLP16] under the same (in fact, weaker) assumption. Our result for DomSet also significantly improves the results in [CL16], but our assumption is stronger.

In fact, we can show even stronger results: the best way to solve Clique and DomSet, even approximately, is to *enumerate all possibilities* in the following sense. Finding a clique of size r can be trivially done in $n^r \operatorname{poly}(n)$ time by checking whether any among all possible $\binom{n}{r} = O(n^r)$ sets of vertices forms a clique. It was known under ETH that this is essentially the best one can do [CHKX06a; CHKX06b]. We show further that this running time is still needed, even when we know that a clique of size much larger than r exists in the graph (e.g., $\mathsf{OPT} \geq 2^{2^r}$), assuming Gap-ETH. Similarly, for DomSet, we can always find a dominating set of size r in $n^r \operatorname{poly}(n)$ time. Under Gap-ETH, we show that there is no better way even when we just want to find a dominating set of size $q \gg r$.

We now give an overview of our techniques. The main challenge in showing our results is that we want them to hold for the case where the optimal solution is *arbitrarily smaller* than the input size. (This is important to get the FPT-inapproximability results.) To this end, (i) reductions cannot blow up the size of the optimal solution by a function of the input size, and (ii) our reductions must start from problems with a large hardness gap while having small OPT. Fortunately, Property (i) holds for the known reductions we employ.

The challenge of (ii) is that existing gap amplifying techniques (e.g., the parallel repetition theorem [Raz98] or the randomized graph product [BS92]), while amplifying the gap to arbitrarily large, cause the input size to be too large that existing OPT reduction techniques (e.g., [CHKX06a; PW10]) cannot be applied efficiently (in particular, in subexponential time). We circumvent this by a step that amplifies the gap and reduce OPT at the same time. In more detail, this step takes a 3SAT formula ϕ as an input and produces a "label cover"[3] instance J (roughly, a bipartite graph with constraints on edges) such that: For any $c > 0$, (i) if ϕ is satisfiable, then J is satisfiable, and (ii) if ϕ is at most 0.99 satisfiable, then less than c-fraction of constraints of J can be satisfied. Moreover, our reduction allows us to "compress" either the the left-hand-side or the right-hand-side vertices to be arbitrarily small. This label cover instance is a starting point for all our problems.

[3]Our problem is an optimization problem on Label Cover instance, with a slightly different objective from the standard Label Cover. Please refer to Section 4 for more detail.

To derive our result for Clique, we would need the left-hand-side to be arbitrarily small while for DomSet, we would need the small right-hand-side.

The left-hand-side vertex compression is similar to the randomized graph product [BS92] and, in fact, the reduction itself has been studied before [Zuc96b; Zuc96a] but in a very different regime of parameters. For a more detailed discussion, please refer to Subsection 4.2.

Once the inapproximability results for label cover problems with small left-hand-side and right-hand-side vertex sets are established, we can simply reduce it to Clique and DomSet using the standard reductions from [FGLSS96] and [Fei98], respectively.

Besides the above results for Clique and DomSet, we also show that no non-trivial FPT-approximation algorithm exists for a few other problems, including Maximum Biclique, the problem of finding maximum subgraphs with hereditary properties (e.g., maximum planar induced subgraph) and Maximum Induced Matching in bipartite graphs. Previously only the exact versions of these problems were known to be W[1]-hard [Lin15; KR00; MS09]. Additionally, we rule out $k^{o(1)}$-FPT-approximation algorithm for Densest k-Subgraph although this ratio does not yet match the trivial $O(k)$-approximation algorithm. Finally, we remark that, while our result for maximum subgraphs with hereditary properties follows from a reduction from Clique, the FPT inapproximability of other problems are shown not through the label cover problems but instead from a modification of the hardness of approximating Densest k-Subgraph in [Man17a].

Previous Works. Our results are based on the method of compressing (or reducing the size of) the optimal solution, which was first introduced by Chen, Huang, Kanj and Xia in [CHKX04] (the journal version appears in [CHKX06a]). Assuming ETH, they showed that finding both Clique and DomSet cannot be solved in time $n^{o(\mathsf{OPT})}$, where n is the number of vertices in an input graph. Later, Pătrașcu and Williams [PW10] applied similar techniques to sharpen the running time lower bound of DomSet to $n^{\mathsf{OPT}-\varepsilon}$, for any constant $\varepsilon > 0$, assuming the *Strong Exponential Time Hypothesis* (SETH). The technique of compressing the optimal solution was also used in hardness of approximation by Hajiaghayi, Khandekar and Kortsarz in [HKK13] and by Bonnet, Lampis and Paschos in [BEKP15]. Our techniques can be seen as introducing gap amplification to the reductions in [CHKX06a]. We emphasize that while [CHKX06a],[PW10],[HKK13] and [BEKP15] (and also the reductions in this paper) are all based on the technique of compressing the optimal solution, Hajiaghayi et al. [HKK13] compress the optimal solution after reducing SAT to the designated problems, i.e., Clique and DomSet. The reductions in [CHKX06a], [PW10], [BEKP15] and in our paper, on the other hand, compress the optimal solution of SAT prior to feeding it to standard reductions (with small adjustment). While this difference does not affect the reduction for Clique, it has a huge effect on DomSet. Specifically, compressing the optimal solution at the post-reduction step results in a huge blow-up because the blow-up in the first step (i.e., from SAT to DomSet) becomes exponential after compressing the optimal solution. Our proof for Clique and the one in [HKK13] bear a similarity in that both apply graph product to amplify approximation hardness. The key difference is that we use randomized graph product instead of the deterministic graph product used in [HKK13].

Very recently, Chen and Lin [CL16] showed that DomSet admits no constant approximation algorithm unless FPT = W[1]. Their hardness result was derived from the seminal result of Lin [Lin15], which shows that the *Maximum k-Intersection* problem (a.k.a, One-side Gap-Biclique) has no FPT approximation algorithm. Furthermore, they showed that, when assuming ETH, their result can be strengthened to rule out $\log^{1/4+\epsilon}(\mathsf{OPT})$ FPT-approximation algorithm, for any constant $\epsilon > 0$. The result of Chen and Lin follows from the W[1]-hardness of Biclique [Lin15] and the proof

of the ETH-hardness of Clique [CHKX04]. Note that while Chen and Lin did not discuss the size of the optimal solution in their paper, the method of compressing the optimal solution was implicitly used there. This is due to the running-time lower bound of Clique that they quoted from [CHKX04].

Our method for proving the FPT inapproximability of DomSet is similar to that in [PW10]. However, the original construction in [PW10] does not require a "partition system". This is because Pătrașcu and Williams reduction starts from SAT, which can be casted as DomSet. In our construction, the reduction starts from an instance of the *Constraint Satisfaction* problem (CSP) that is more general than SAT (because of the gap-amplification step) and hence requires the construction of a partition system. (Note that the partition system has been used in standard hardness reductions for DomSet [LY94; Fei98].)

We remark that our proof does not imply FPT-inapproximability for DomSet under ETH whereas Chen and Lin were able to prove the inapproximability result under ETH because their reduction can be applied directly to SAT via the result in [CHKX06a]. If ones introduced the Gap-ETH to the previous works, then the proofs in [CHKX06a; HKK13; BEKP15] yield the constant FPT-inapproximability of Clique, and the proof in [CHKX06a] yields the constant FPT-inapproximability of DomSet.

The summaries of previous works on Clique and DomSet are presented in Table 1.

Summary of Works on Clique

Inapprox Factor	Running Time Lower Bound	Assumption	References
any constant	$t(OPT) \cdot n^{o(\mathsf{OPT})}$	ETH + LPCP	[BEKP15]
$\mathsf{OPT}^{1-\epsilon}$	$\exp(\mathsf{OPT}^{\rho(\epsilon)})$	ETH	[CHK13]
$1/(1-\epsilon)$	$\exp(\exp(\mathsf{OPT}^{\rho(\epsilon)}))$[4]	ETH	[HKK13]
No $\omega(\mathsf{OPT})$	$t(OPT) \cdot n^{o(\mathsf{OPT})}$	Gap-ETH	This paper

Summary of Works on DomSet

Inapprox Factor	Running Time Lower Bound	Assumption	References
$\mathsf{OPT}^{1-\gamma}$	$\exp(\mathsf{OPT}^{1-\rho(\gamma)})$	ETH	[CHK13]
$(\log \mathsf{OPT})^{\delta}$	$\exp(\exp((\log \mathsf{OPT})^{\delta-1}))$	ETH + PGC	[HKK13]
any constant	$t(OPT) \cdot n^{O(1)}$ (i.e., no FPT)	W[1] $\neq$ FPT	[CL16]
$(\log \mathsf{OPT})^{1/4+\epsilon}$	$t(OPT) \cdot n^{o(\sqrt{\mathsf{OPT}})}$	ETH	[CL16]+[CHKX06a]
$f(\mathsf{OPT})$	$t(OPT) \cdot n^{o(\mathsf{OPT})}$	Gap-ETH	This paper

Table 1: The summaries of previous works on Clique and DomSet. Here t denotes any computable function $t : \mathbb{N} \to \mathbb{N}$, ϵ denotes any constant $0 < \varepsilon < 1$, γ denotes some constant $0 < \epsilon < 1$, ρ denotes some non-decreasing function $\rho : (0,1) \to (0,1)$, δ denotes some constant $\delta > 1$. PGC and LPCP stands for the *Projection Game Conjecture* [Mos15], and the *Linear-Size PCP Conjecture* [BEKP15], respectively.

Follow-up works. In a follow-up work by Karthik, Laekhanukit and Manurangsi [KLM17], the authors show that it is W[1]-hard to non-trivially approximate DomSet. Furthermore, they are able to bypass Gap-ETH and show the same result as ours, i.e., that enumeration is essentially the best

[4]Constant FPT-inapproximability of Clique under ETH is claimed in [HKK13] (arXiv version). However, as we investigated, the Gap-ETH is assumed there.

possible for DomSet under ETH. Finally, under SETH and the k-Sum Hypothesis, stronger running time lower bounds are shown there. We stress here that their results apply only to DomSet but not to Clique nor other problems in this paper (e.g., Biclique, Densest k-Subgraph). Their proof builds on the insights from our work; in particular, our reduction from a parameterized version of label cover to DomSet (Theorem 5.4) is needed there. The contrast between our proof for DomSet and the proof in [KLM17] is that, while we obtain a gap in the label cover problem from Gap-ETH, [KLM17] takes a different route and arrives at such a gap by designing specific communication protocols for certain communication problems, generalizing connections between communication complexity and fine-grained hardness of approximation pioneered in [ARW17].

Other Related Works. All problems considered in this work are also well-studied in terms of hardness of approximation beyond the aforementioned parameterized regimes; indeed many techniques used here are borrowed from or inspired by the non-parameterized settings.

Maximum Clique. Maximum Clique is arguably the first natural combinatorial optimization problem studied in the context of hardness of approximation; in a seminal work of Feige, Goldwasser, Lovász, Safra and Szegedy (henceforth FGLSS) [FGLSS96], a connection was made between interactive proofs and hardness of approximating Clique. This connection paves the way for later works on Clique and other developments in the field of hardness of approximations; indeed, the FGLSS reduction will serve as part of our proof as well. The FGLSS reduction, together with the PCP theorem [AS98; ALMSS98] and gap amplification via randomized graph products [BS92], immediately implies n^{ε} ratio inapproximability of Clique for some constant $\varepsilon > 0$ under the assumption that NP$\subseteq$ BPP. Following Feige et al.'s work, there had been a long line of research on approximability of Clique [BGLR93; FK00; BGS98; BS94], which culminated in Håstad's work [Hås96]. In [Hås96], it was shown that Clique cannot be approximated to within a factor of $n^{1-\varepsilon}$ in polynomial time unless NP$\subseteq$ ZPP; this was later derandomized by Zuckerman [Zuc07] via an efficient construction of *dispersers* with certain parameters, thus implying $n^{1-\varepsilon}$ approximation hardness under P $\neq$ NP. Since then, better inapproximability ratios are known [EH00; Kho01; KP06], with the best ratio being $n/2^{(\log n)^{3/4+\varepsilon}}$ for every $\varepsilon > 0$ (assuming NP$\not\subseteq$ BPTIME$(2^{(\log n)^{O(1)}})$) due to Khot and Ponnuswami [KP06]. We note here that the best known polynomial time algorithm for Clique achieves $O\left(\frac{n(\log\log n)^2}{(\log n)^3}\right)$-approximation for the problem [Fei04].

Set Cover. Minimum Set Cover, which is equivalent to DomSet, is also among the first problems studied in hardness of approximation. Lund and Yannakakis proved that, unless NP$\subseteq$ DTIME$(2^{(\log n)^{O(1)}})$, SetCov cannot be efficiently approximated to within $c \log n$ factor of the optimum for some constant $c > 0$ [LY94]. Not long after, Feige [Fei98] both improved the approximation ratio and weaken the assumption by showing an $(1-\varepsilon)\ln n$-ratio inapproximability for every $\varepsilon > 0$ assuming only that NP$\not\subseteq$ DTIME$(n^{O(\log\log n)})$. Recently, a similar inapproximability has been achieved under the weaker NP$\not\subseteq$ P assumption [Mos15; DS14]. Since a simple greedy algorithm is known to yield $(\ln n + 1)$-approximation for SetCov [Chv79], the aforementioned hardness result is essentially tight. A common feature in all previous works on hardness of SetCov [LY94; Fei98; Mos15] is that the constructions involve composing certain variants of CSPs with partition systems. As touched upon briefly earlier, our construction will also follow this approach; for the exact definition of CSPs and the partition system used in our work, please refer to Subsection 5.2.2.

Maximum Subgraph with Hereditary Properties. The complexity of finding and approximating maximum subgraph with hereditary properties have also been studied since the 1980s [LY80; LY93; FK05]; specifically, Feige and Kogan showed that, for every non-trivial property Π (i.e., Π such that infinite many subgraphs satisfy Π and infinitely many subgraphs do not satisfy Π), the

problem is hard to approximate to within $n^{1-\varepsilon}$ factor for every $\varepsilon > 0$ unless NP$\subseteq$ ZPP [FK05]. We also note that non-trivial approximation algorithms for the problem are known; for instance, when the property fails for some clique or some independent set, a polynomial time $O\left(\frac{n(\log\log n)^2}{(\log n)^2}\right)$-approximation algorithm is known [Hal00].

Maximum Balanced Biclique. While the Maximum Balanced Biclique problem bears a strong resemblance to the Maximum Clique Problem, inapproximability of the latter cannot be directly translated to that of the former; in fact, despite numerous attempts, not even constant factor NP-hardness of approximation of the Maximum Balanced Biclique problem is known. Fortunately, under stronger assumptions, hardness of approximation for the problem is known: n^{ε}-factor hardness of approximation is known under Feige's random 3SAT hypothesis [Fei02] or NP$\not\subseteq \bigcap_{\varepsilon>0}$BPTIME$(2^{n^{\varepsilon}})$ [Kho06], and $n^{1-\varepsilon}$-factor hardness of approximation is known under strengthening of the Unique Games Conjecture [BGHKK16; Man17b]. To the best of our knowledge, no non-trivial approximation algorithm for the problem is known.

Densest k-Subgraph. The Densest k-Subgraph problem has received considerable attention from the approximation algorithm community [KP93; FKP01; BCCFV10]; the best known polynomial time algorithm due to Bhaskara et al. [BCCFV10] achieves $O(n^{1/4+\varepsilon})$-approximation for every $\varepsilon > 0$. On the other hand, similar to Biclique, NP-hardness of approximating Densest k-Subgraph, even to some constant ratio, has so far eluded researchers. Nevertheless, in the same works that provide hardness results for Biclique [Fei02; Kho06], DkS is shown to be hard to approximate to some constant factor under random 3-SAT hypothesis or NP$\not\subseteq \bigcap_{\varepsilon>0}$BPTIME$(2^{n^{\varepsilon}})$. Furthermore, $2^{\Omega(\log^{2/3} n)}$-factor inapproximability is known under the planted clique hypothesis [AAMMW11] and, under ETH (respectively, Gap-ETH), $n^{1/\operatorname{poly}\log\log n}$ (respectively, $n^{o(1)}$) factor inapproximabilities are known [Man17a]. (See also [BKRW17] in which a constant ratio ETH-hardness of approximating DkS was shown.) In addition to these hardness results, polynomial ratio integrality gaps for strong LP and SDP relaxations of the problem are also known [BCVGZ12; Man15; CMMV17].

Maximum Induced Matching on Bipartite Graphs. The problem was proved to be NP-hard independently by Stockmeyer and Vazirani [SV82] and Cameron [Cam89]. The approximability of the problem was first studied by Duckworth et al. [DMZ05] who showed that the problem is APX-hard, even on bipartite graphs of degree three. Elbassioni et al. [ERRS09] then showed that the problem is hard to approximate to within $n^{1/3-\varepsilon}$ factor for every $\varepsilon > 0$, unless NP$\subseteq$ ZPP. Chalermsook et al. [CLN13a] later improved the ratio to $n^{1-\varepsilon}$ for every $\varepsilon > 0$.

Organization. We define basic notations in Section 2. In Section 3, we define the notion of inherently enumerative, which captures the fact that nothing better than enumerating all possibilities can be done. We show that a problem admits no non-trivial FPT-approximation algorithm by showing that it is inherently enumerative. In Section 4, we define and prove results about our intermediate problems on label cover instances. Finally, in Section 5 we derive results for Clique, DomSet, and other problems.

2 Preliminaries

We use standard terminology. For any graph G, we denote by $V(G)$ and $E(G)$ the vertex and edge sets of G, respectively. For each vertex $u \in V(G)$, we denote the set of its neighbors by $N_G(v)$; when the graph G is clear from the context, we sometimes drop it from the notation. A *clique* of G is a complete subgraph of G. Sometime we refer to a clique as a subset $S \subseteq V(G)$

such that there is an edge joining every pair of vertices in S. A *biclique* of G is a balanced complete bipartite subgraph of G (i.e., the graph $K_{k,k}$). By k-biclique, we mean the graph $K_{k,k}$ (i.e., the number of vertices in each partition is k). An *independent set* of G is a subset of vertices $S \subseteq V(G)$ such there is no edge joining any pair of vertices in S. A *dominating set* of G is a subset of vertices $S \subseteq V(G)$ such that every vertex in G is either in S or has a neighbor in S. The *clique number* (respectively, *independent number*) of G is the size of the largest clique (respectively, independent set) in G. The *biclique number* of G is the largest integer k such that G contains $K_{k,k}$ as a subgraph. The *domination number* of G is defined similarly as the size of the smallest dominating set in G. The clique, independent and domination numbers of G are usually denoted by $\omega(G)$, $\alpha(G)$ and $\gamma(G)$, respectively. However, in this paper, we will refer to these numbers by $\mathsf{Clique}(G), \mathsf{MIS}(G), \mathsf{DomSet}(G)$. Additionally, we denote the biclique number of G by $\mathsf{Biclique}(G)$

2.1 FPT Approximation

Let us start by formalizing the the notation of optimization problems; here we follow the notation due to Chen et al. [CGG06]. An *optimization problem* Π is defined by three components: (1) for each input instance I of Π, a set of valid solutions of I denoted by $\mathsf{SOL}_\Pi(I)$, (2) for each instance I of Π and each $y \in \mathsf{SOL}_\Pi(I)$, the cost of y with respect to I denoted by $\mathsf{COST}_\Pi(I, y)$, and (3) the goal of the problem $\mathsf{GOAL}_\Pi \in \{\min, \max\}$ which specifies whether Π is a minimization or maximization problem. Throughout this work, we will assume that $\mathsf{COST}_\Pi(I, y)$ can be computed in time $|I|^{O(1)}$. Finally, we denote by $\mathsf{OPT}_\Pi(I)$ the optimal value of each instance I, i.e., $\mathsf{OPT}_\Pi(I) = \mathsf{GOAL}_\Pi\ \mathsf{COST}(I, y)$ where y is taken over $\mathsf{SOL}_\Pi(I)$.

We now continue on to define parameterized approximation algorithms. While our discussion so far has been on optimization problems, we will instead work with "gap versions" of these problems. Roughly speaking, for a maximization problem Π, the gap version of Π takes in an additional input k and the goal is to decide whether $\mathsf{OPT}_\Pi(I) \geq k$ or $\mathsf{OPT}_\Pi(I) < k/f(k)$. As we will elaborate below, the gap versions are weaker (i.e., easier) than the optimization versions and, hence, our impossibility results for gap versions translate to those of optimization versions as well.

Definition 2.1 (FPT gap approximation)**.** *For any optimization problem Π and any computable function $f : \mathbb{N} \to [1, \infty)$, an algorithm $\mathbb{A}$, which takes as input an instance I of Π and a positive integer k, is said to be an* f-FPT gap approximation algorithm *for Π if the following conditions hold on every input (I, k):*

- *$\mathbb{A}$ runs in time $t(k) \cdot |I|^{O(1)}$ for some computable function $t : \mathbb{N} \to \mathbb{N}$.*
- *If $\mathsf{GOAL}_\Pi = \max$, $\mathbb{A}$ must output 1 if $\mathsf{OPT}_\Pi(I) \geq k$ and output 0 if $\mathsf{OPT}_\Pi(I) < k/f(k)$.*
  If $\mathsf{GOAL}_\Pi = \min$, $\mathbb{A}$ must output 1 if $\mathsf{OPT}_\Pi(I) \leq k$ and output 0 if $\mathsf{OPT}_\Pi(I) > k \cdot f(k)$.

Π is said to be f-FPT gap approximable *if there is an f-FPT gap approximation algorithm for Π.*

Next, we formalize the concept of *totally FPT inapproximable*, which encapsulates the non-existence of non-trivial FPT approximations discussed earlier in the introduction.

Definition 2.2. *A minimization problem Π is said to be* totally FPT inapproximable *if, for every computable function $f : \mathbb{N} \to [1, \infty)$, Π is not f-FPT gap approximable.*

A maximization problem Π is said to be totally FPT inapproximable *if, for every computable function $f : \mathbb{N} \to [1, \infty)$ such that $f(k) = o(k)$ (i.e., $\lim_{k\to\infty} k/f(k) = \infty$), Π is not f-FPT gap approximable.*

With the exception of Densest k-Subgraph, every problem considered in this work will be shown to be totally FPT inapproximable. To this end, we remark that totally FPT inapproximable as defined above through gap problems imply the non-existence of non-trivial FPT approximation algorithm that was discussed in the introduction. These implications are stated more precisely in the two propositions below; their proofs are given in Appendix A.

Proposition 2.3. *Let Π be any minimization problem. Then (1) implies (2) where (1) and (2) are as defined below.*

(1) Π is totally FPT inapproximable.

(2) For all computable functions $t : \mathbb{N} \to \mathbb{N}$ and $f : \mathbb{N} \to [1, \infty)$, there is no algorithm that, on every instance I of Π, runs in time $t(\mathsf{OPT}_\Pi(I)) \cdot |I|^{O(1)}$ and outputs a solution $y \in \mathsf{SOL}_\Pi(I)$ such that $\mathsf{COST}_\Pi(I, y) \leq \mathsf{OPT}_\Pi(I) \cdot f(\mathsf{OPT}_\Pi(I))$.

Proposition 2.4. *Let Π be any maximization problem. Then (1) implies (2) where (1) and (2) are as defined below.*

(1) Π is totally FPT inapproximable.

(2) For all computable functions $t : \mathbb{N} \to \mathbb{N}$ and $f : \mathbb{N} \to [1, \infty)$ such that $f(k) = o(k)$ and $k/f(k)$ is non-decreasing, there is no algorithm that, on every instance I of Π, runs in time $t(\mathsf{OPT}_\Pi(I)) \cdot |I|^{O(1)}$ and outputs a solution $y \in \mathsf{SOL}_\Pi(I)$ such that $\mathsf{COST}_\Pi(I, y) \geq \mathsf{OPT}_\Pi(I)/f(\mathsf{OPT}_\Pi(I))$.

2.2 List of Problems

We will now list the problems studied in this work. While all the problems here can be defined in terms of optimization problems as discussed in the previous subsection, we will omit the terms $\mathsf{SOL}, \mathsf{COST}$ and GOAL whenever they are clear from the context.

The Maximum Clique Problem (Clique). In k-Clique, we are given a graph G together with an integer k, and the goal is to decide whether G has a clique of size k. The maximization version of k-Clique, called Max-Clique or simply Clique, asks to compute the maximum size of a clique in G.

The problem that is (computationally) equivalent to Clique is the *maximum independent set* problem (MIS) which asks to compute the size of the maximum independent set in G. The two problems are equivalent since any clique in G is an independent set in the complement graph $\bar{G}$.

The Minimum Dominating Set Problem (DomSet). In k-DomSet, we are given a graph G together with an integer k, and the goal is to decide whether G has a dominating set of size k. The minimization version of k-DomSet, called Min-DomSet or simply DomSet, asks to compute the size of the minimum dominating set in G.

The problem that is equivalent to DomSet is the *minimum set cover* problem (SetCov): Given a universe $\mathcal{U}$ of n elements and a collection $\mathcal{S}$ of m subsets $S_1, \ldots, S_m \subseteq \mathcal{U}$, the goal is to find the minimum number of subsets of $\mathcal{S}$ whose union equals $\mathcal{U}$. It is a standard fact that DomSet is equivalent to SetCov. See Appendix D for more detail.

Maximum Induced Subgraph with Hereditary Properties: A *property* Π is simply a subset of all graphs. We say that Π is a *hereditary property* if whenever $G \in \Pi$, all induced subgraphs of G are in Π. The Maximum Induced Subgraph problem with Property Π asks for a maximum

cardinality set $S \subseteq V(G)$ such that $G[S] \in \Pi$. Here $G[S]$ denotes the subgraph of G induced on S. Notice that both Clique and MIS belong to this class of problems. For more discussions on problems that belong to this class, see Appendix D.

Maximum Induced Matching on Bipartite Graphs: An induced matching $\mathcal{M}$ of a graph $G = (V, E)$ is a subset of edges $\{(u_1, v_1), \ldots, (u_{|\mathcal{M}|}, v_{|\mathcal{M}|})\}$ such that there is no cross edge, i.e., $(u_i, u_j), (v_i, v_j), (u_i, v_j) \notin E$ for all $i \neq j$. The induced matching number $\mathsf{IM}(G)$ of graph G is simply the maximum value of $|\mathcal{M}|$ among all induced matchings $\mathcal{M}$'s of G. In this work, we will be interested in the problem of approximating $\mathsf{IM}(G)$ in bipartite graphs; this is because, for general graphs, the problem is as hard to approximate as Clique. (See Appendix D for more details.)

Maximum Balanced Biclique (Biclique). In k-Biclique, we are given a bipartite graph G together with an integer k. The goal is to decide whether G contains a complete bipartite subgraph (biclique) with k vertices on each side. In other words, we are asked to decide whether G contains $K_{k,k}$ as a subgraph. The maximization version of Biclique, called Maximum Balanced Biclique, asks to compute the maximum size of a balanced biclique in G.

Densest k-Subgraph (DkS). In the Densest k-Subgraph problem, we are given an integer k and a graph $G = (V, E)$. The goal is to find a subset $S \subseteq V$ of k vertices that induces maximum number of edges. For convenience, we define density of an induced subgraph $G[S]$ to be $\mathsf{Den}(G[S]) \triangleq \frac{E(G[S])}{\binom{|S|}{2}} \in [0, 1]$ and we define the optimal density of DkS to be $\mathsf{Den}_k(G) = \max_{S \subseteq V, |S|=k} \mathsf{Den}(S)$.

2.3 Gap Exponential Time Hypothesis

Our results are based on the Gap Exponential Time Hypothesis (Gap-ETH). Before we state the hypothesis, let us recall the definition of 3-SAT. In q-SAT, we are given a CNF formula ϕ in which each clause consists of at most q literals, and the goal is to decide whether ϕ is satisfiable.

Max q-SAT is a maximization version of q-SAT which asks to compute the maximum number of clauses in ϕ that can be simultaneously satisfied. We will abuse q-SAT to mean Max q-SAT, and for a formula ϕ, we use $\mathsf{SAT}(\phi)$ to denote the maximum number of clauses satisfied by any assignment.

The Gap Exponential Time Hypothesis can now be stated in terms of SAT as follows.

Conjecture 2.5 ((randomized) Gap Exponential-Time Hypothesis (Gap-ETH) [Din16; MR16])**.** *For some constant $\delta, \epsilon > 0$, no algorithm can, given a 3-SAT formula ϕ on n variables and $m = O(n)$ clauses, distinguishes between the following cases correctly with probability $\geq 2/3$ in $O(2^{\delta n})$ time:*

- $\mathsf{SAT}(\phi) = m$ *and*
- $\mathsf{SAT}(\phi) < (1 - \epsilon)m$.

Note that the case where $\epsilon = 1/m$ (that is, the algorithm only needs to distinguish between the cases that $\mathsf{SAT}(\phi) = m$ and $\mathsf{SAT}(\phi) < m$) is known as ETH [IPZ01]. Another related conjecture is the strengthened version of ETH is called *the Strong Exponential-Time Hypothesis* (SETH) [IP01]: for any $\epsilon > 0$, there is an integer $k \geq 3$ such that there is no $2^{(1-\epsilon)n}$-time algorithm for k-SAT. Gap-ETH of course implies ETH, but, to the best of our knowledge, no formal relationship is known between Gap-ETH and SETH. While Gap-ETH may seem strong due to the gap between the two cases, there are evidences suggesting that it may indeed be true, or, at the very least, refuting it is beyond the reach of our current techniques. We discuss some of these evidences in Appendix F.

While Gap-ETH as stated above rules out not only deterministic but also randomized algorithms, the deterministic version of Gap-ETH suffices for some of our results, including inapproximability of Clique and DomSet. The reduction for DomSet as stated below will already be deterministic, but the reduction for Clique will be randomized. However, it can be easily derandomized and we sketch the idea behind this in in Subsection 4.2.1. Note that, on the other hand, we do not know how to derandomize some of our other results, including those of Biclique and DkS.

3 FPT Inapproximability via Inherently Enumerative Concept

Throughout the paper, we will prove FPT inapproximability through the concept of *inherently enumerative* problems, which will be formalized shortly.

To motivate the concept, note that all problems Π considered in this paper admit an exact algorithm that runs in time[5] $O^\star(|I|^{\mathsf{OPT}_\Pi(I)})$ For instance, to find a clique of size k in G, one can enumerate all $\binom{|V(G)|}{k} = |V(G)|^{O(k)}$ possibilities[6]. For many W[1]-hard problems (e.g., Clique), this running time is nearly the best possible assuming ETH: Any algorithm that finds a k-clique in time $|V(G)|^{o(k)}$ would break ETH. In the light of such result, it is natural to ask the following question.

Assume that $\mathsf{Clique}(G) \geq 2^{2^k}$, can we find a clique of size k in time $|V(G)|^{o(k)}$?

In other words, can we exploit a prior knowledge that there is a clique of size much larger than k to help us find a k-clique faster? Roughly speaking, we will show later that, assuming Gap-ETH, the answer of this question is also negative, even when 2^{2^k} is replaced by any constant independent of k. This is encapsulated in the concept of inherently enumerative as defined below.

Definition 3.1 (Inherently Enumerative)**.** *A problem* Π *is said to be* inherently enumerative *if there exist constants* $\delta, r_0 > 0$ *such that, for any integers* $q \geq r \geq r_0$*, no algorithm can decide, on every input instance* I *of* Π*, whether (i)* $\mathsf{OPT}_\Pi(I) < r$ *or (ii)* $\mathsf{OPT}_\Pi(I) \geq q$ *in time*[7] $O_{q,r}(|I|^{\delta r})$*.*

While we will show that Clique and DomSet are inherently enumerative, we cannot do the same for some other problems, such as Biclique. Even for the exact version of Biclique, the best running time lower bound known is only $|V(G)|^{\Omega(\sqrt{k})}$ [Lin15] assuming ETH. In order to succinctly categorize such lower bounds, we define a similar but weaker notation of *weakly* inherently enumerative:

Definition 3.2 (Weakly Inherently Enumerative)**.** *For any function* $\beta = \omega(1)$ *(i.e.,* $\lim_{r\to\infty} \beta(r) = \infty$*), a problem* Π *is said to be* β-weakly inherently enumerative *if there exists a constant* $r_0 > 0$ *such that, for any integers* $q \geq r \geq r_0$*, no algorithm can decide, on every input instance* I *of* Π*, whether (i)* $\mathsf{OPT}_\Pi(I) < r$ *or (ii)* $\mathsf{OPT}_\Pi(I) \geq q$ *in time* $O_{q,r}(|I|^{\beta(r)})$*.*

Π *is said to be* weakly inherently enumerative *if it is* β*-weakly inherently enumerative for some* $\beta = \omega(1)$*.*

It follows from the definitions that any inherently enumerative problem is also weakly inherently enumerative. As stated earlier, we will prove total FPT inapproximability through inherently enumerative; the proposition below formally establishes a connection between the two.

[5] Recall that $O^\star(\cdot)$ hides terms that are polynomial in the input size.

[6] A faster algorithm runs in time $|V(G)|^{\omega k/3}$ can be done by a reduction to matrix multiplication.

[7] $O_{q,r}(\cdot)$ here and in Definition 3.2 hides any multiplicative term that is a function of q and r.

Proposition 3.3. *If* Π *is weakly inherently enumerative, then* Π *is totally FPT inapproximable.*

Proof. We first consider maximization problems. We will prove the contrapositive of the statement. Assume that a maximization problem Π is not totally FPT inapproximable, i.e., Π admits an f-FPT gap approximation algorithm $\mathbb{A}$ for some computable function f such that $\lim_{k\to\infty} k/f(k) = \infty$. Suppose that the running time of $\mathbb{A}$ on every input (I, k) is $t(k) \cdot |I|^D$ for some constant D and some function t. We will show that Π is not weakly inherently enumerative.

Let $r_0 > 0$ be any constant and let $\beta : \mathbb{N} \to \mathbb{R}^+$ be any function such that $\beta = \omega(1)$. Let r be the smallest integer such that $r > r_0$ and $\beta(r) \geq D$ and let q be the smallest integer such that $q/f(q) > r$. Note that r and q exists since $\lim_{r\to\infty} \beta(r) = \infty$ and $\lim_{q\to\infty} q/f(q) = \infty$.

Given any instance I of Π. From the definition of f-FPT gap approximation algorithms (Definition 2.1) and from the fact that $q/f(q) > r$, $\mathbb{A}$ on the input (I, q) can distinguish between $\mathsf{OPT}_\Pi(I) \geq q$ and $\mathsf{OPT}_\Pi(I) < r$ in $t(q) \cdot |I|^D \leq t(q) \cdot |I|^{\beta(r)} = O_{q,r}(|I|^{\beta(r)})$ time. Hence, Π is not weakly inherently enumerative, concluding our proof for maximization problems.

For any minimization problem Π, assume again that Π is not totally FPT inapproximable, i.e., Π admits an f-FPT gap approximation algorithm $\mathbb{A}$ for some computable function f. Suppose that the running time of $\mathbb{A}$ on every input (I, k) is $t(k) \cdot |I|^D$ for some constant D.

Let $r_0 > 0$ be any constant and let $\beta : \mathbb{N} \to \mathbb{R}^+$ be any function such that $\beta = \omega(1)$. Let r be the smallest integer such that $r > r_0$ and $\beta(r) \geq D$ and let $q = \lceil r \cdot f(r) \rceil$.

Given any instance I of Π. From definition of f-FPT gap approximation algorithms and from $q \geq r \cdot f(r)$, $\mathbb{A}$ on the input (I, r) can distinguish between $\mathsf{OPT}_\Pi(I) \geq q$ and $\mathsf{OPT}_\Pi(I) < r$ in $t(r) \cdot |I|^D \leq t(r) \cdot |I|^{\beta(r)} = O_{q,r}(|I|^{\beta(r)})$ time. Hence, Π is not weakly inherently enumerative. □

An important tool in almost any branch of complexity theory, including parameterized complexity, is a notion of reductions. For the purpose of facilitating proofs of totally FPT inapproximability, we define the following reduction, which we call *FPT gap reductions*.

Definition 3.4 (FPT gap reduction)**.** *For any functions* $f, g = \omega(1)$*, a problem* Π_0 *is said to be* (f, g)-FPT gap reducible *to a problem* Π_1 *if there exists an algorithm* $\mathbb{A}$ *which takes an instance* I_0 *of* Π_0 *and integers* q, r *and then produces an instance* I_1 *of* Π_1 *such that the following conditions hold.*

- $\mathbb{A}$ *runs in time* $t(q, r) \cdot |I_0|^{O(1)}$ *for some computable function* $t : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$*.*
- *For every positive integer* q*, if* $\mathsf{OPT}_{\Pi_0}(I_0) \geq q$*, then* $\mathsf{OPT}_{\Pi_1}(I_1) \geq f(q)$*.*
- *For every positive integer* r*, if* $\mathsf{OPT}_{\Pi_0}(I_0) < g(r)$*, then* $\mathsf{OPT}_{\Pi_1}(I_1) < r$*.*

It is not hard to see that FPT gap reduction indeed preserves totally FPT inapproximability, as formalized in Proposition 3.5 below. The proof of the proposition can be found in Appendix B.

Proposition 3.5. *If a problem* Π_0 *is (i)* (f, g)*-FPT gap reducible to* Π_1 *for some computable non-decreasing functions* $f, g = \omega(1)$*, and (ii) totally FPT inapproximable, then* Π_1 *is also totally FPT inapproximable.*

As stated earlier, we mainly work with inherently enumerative concepts instead of working directly with totally FPT inapproximability; indeed, we will never use the above proposition and we alternatively use FPT gap reductions to prove that problems are weakly inherently enumerative For this purpose, we will need the following proposition.

Proposition 3.6. *If a problem Π_0 is (i) (f, g)-FPT gap reducible to Π_1 and (ii) β-weakly inherently enumerative for some $f, g, \beta = \omega(1)$, then Π_1 is $\Omega(\beta \circ g)$-weakly inherently enumerative.*

Proof. We assume that (i) holds, and will show that if the "then" part does not hold, then (ii) also does not hold. Recall from Definition 3.4 that (i) implies that there exists $C, D > 0$ such that the reduction from Π_0 (with parameters q and r) to Π_1 takes $O_{q,r}(|I_0|^C)$ time and always output an instance I_1 of size at most $O_{q,r}(|I_0|^D)$ on every input instance I_0. Now assume that the "then" part does *not* hold, in particular Π_1 is *not* $(\beta \circ g)/D$-weakly inherently enumerative. We will show the following claim which says that (ii) does not hold (by Definition 3.2).

Claim 3.7. *For every $r_0 > 0$, there exists $q \geq r \geq r_0$ and an $O_{q,r}(|I_0|^{\beta(r)})$-time algorithm $\mathbb{B}$ that can, on every input instance I_0 of Π_0, distinguish between $\mathsf{OPT}_{\Pi_0}(I_0) \geq q$ and $\mathsf{OPT}_{\Pi_0}(I_0) < r$.*

We now prove the claim. Consider any r_0. Since $\beta, g = \omega(1)$, there exists r_0' such that $g(r') \geq r_0$ and $\beta(r') \geq C$, for all $r' \geq r_0'$. From the assumption that Π_1 is not $(\beta \circ g)/D$-weakly inherently enumerative, there exist $q' \geq r' \geq r_0'$ such that there is an $O_{q',r'}(|I_1|^{\beta(g(r'))/D})$-time algorithm $\mathbb{A}$ that can, on every input instance I_1 of Π_1, distinguish between $\mathsf{OPT}_{\Pi_1}(I_1) \geq q'$ and $\mathsf{OPT}_{\Pi_1}(I_1) < r'$.

Let $r = g(r')$, and let q be the smallest integer such that $f(q) \geq q'$ and $q \geq r$. Note that q exists since $\lim_{q\to\infty} f(q) = \infty$, and that $r \geq r_0$. We use $\mathbb{A}$ and the reduction to build an algorithm $\mathbb{B}$ as follows. On input I_0, the algorithm $\mathbb{B}$ runs the reduction on I_0 and the previously defined q, r. Let us call the output of the reduction I_1. The algorithm $\mathbb{B}$ then runs $\mathbb{A}$ on the input (I_1, q', r') and outputs accordingly, i.e., if $\mathbb{A}$ says that $\mathsf{OPT}_{\Pi_1}(I_1) \geq q'$, then $\mathbb{B}$ outputs $\mathsf{OPT}_{\Pi_0}(I_0) \geq q$, and, otherwise, if $\mathbb{A}$ says that $\mathsf{OPT}_{\Pi_1}(I_1) < r'$, then $\mathbb{B}$ outputs $\mathsf{OPT}_{\Pi_0}(I_0) < r$.

Now we show that $\mathbb{B}$ can distinguish whether $\mathsf{OPT}_{\Pi_0}(I_0) \geq q$ or $\mathsf{OPT}_{\Pi_1}(I_1) < r$ as desired by the claim: From our choice of q, if $\mathsf{OPT}_{\Pi_0}(I_0) \geq q$, then $\mathsf{OPT}_{\Pi_1}(I_1) \geq f(q) \geq q'$. Similarly, from our choice of $r = g(r')$, if $\mathsf{OPT}_{\Pi_0}(I_0) < r$, then $\mathsf{OPT}_{\Pi_1}(I_1) < r'$. Since $\mathbb{A}$ can distinguish between the two cases, $\mathbb{B}$ can distinguish between the two cases as well.

The total running time of $\mathbb{B}$ is $O_{q,r}(|I_0|^C) + O_{q',r'}(|I_1|^{\beta(g(r'))/D})$ (the first term is for running the reduction). Since I_1 is of size at most $O_{q,r}(|I_0|^D)$, $\beta(r) \geq C$, and q' and r' depend only on q and r, the running time can be bounded by $O_{q,r}(|I_0|^{\beta(r)})$ as desired. □

4 Covering Problems on Label Cover Instances

In this section, we give intermediate results for the lower bounds on the running time of approximating variants of the *label cover* problem, which will be the source of our inapproximability results for Clique and DomSet.

4.1 Problems and Results

Label cover instance: A label cover instance Γ consists of $(G, \Sigma_U, \Sigma_V, \Pi)$, where

- $G = (U, V, E)$ is a bipartite graph between vertex sets U and V and an edge set E,
- Σ_U and Σ_V are sets of *alphabets* to be assigned to vertices in U and V, respectively, and
- $\Pi = \{\Pi_e\}_{e\in E}$ is a set of *constraints* $\Pi_e \subseteq \Sigma_U \times \Sigma_V$.

We say that Π (or Γ) has the *projection property* if for every edge $uv \in E$ (where $u \in U$ and $V \in v$) and every $\alpha \in \Sigma_U$, there is exactly one $\beta \in \Sigma_V$ such that $(\alpha, \beta) \in \Pi_{uv}$.

We will define two combinatorial optimization problems on an instance of the label cover problem. These two problems are defined on the same instance as the standard label cover problem. We will briefly discuss how our problems differ from the standard one.

Max-Cover Problem: A *labeling* of the graph, is a pair of mappings $\sigma_U : U \to \Sigma_U$ and $\sigma_V : V \to \Sigma_V$. We say that a labeling (σ_U, σ_V) *covers* edge uv if $(\sigma_U(u), \sigma_V(v)) \in \Pi_{uv}$. We say that a labeling covers a vertex u if it covers every edge incident to u. For any label cover instance Γ, let $\mathsf{MaxCov}(\Gamma)$ denote the maximum number of vertices in U that can be covered by a labeling; i.e.

$$\mathsf{MaxCov}(\Gamma) := \max_{\sigma_U : U \to \Sigma_U,\ \sigma_V : V \to \Sigma_V} |\{u \in U \mid (\sigma_U, \sigma_V) \text{ covers } u\}|.$$

The goal of the Max-Cover problem is to compute $\mathsf{MaxCov}(\Gamma)$. We remark that the standard label cover problem (e.g., [WS11]) would try to maximize the number of covered *edges*, as opposed to our Max-Cover problem, which seeks to maximize the number of covered *vertices*.

Min-Label Problem: A *multi-labeling* of the graph, is a pair of mappings $\sigma_U : U \to \Sigma_U$ and $\hat{\sigma}_V : V \to 2^{\Sigma_V}$. We say that $(\sigma_U, \hat{\sigma}_V)$ *covers* an edge uv, if there exists $\beta \in \hat{\sigma}_V(v)$ such that $(\sigma(u), \beta) \in \Pi_{uv}$. For any label cover instance Γ, let $\mathsf{MinLab}(\Gamma)$ denote the minimum number of labels needed to assign to vertices in V in order to cover *all* vertices in U, i.e.,

$$\mathsf{MinLab}(\Gamma) := \min_{(\sigma_U, \hat{\sigma}_V)} \sum_{v \in V} |\hat{\sigma}_V(v)|$$

where the minimization is over multi-labelings $(\sigma_U, \hat{\sigma}_V)$ that covers every edge in G.

It is worth noting that, in MinLab, we are allowed to assign multiple labels to vertices in V whereas each vertex in U must be assigned a unique label. This makes MinLab different from the problem known in the literature as MinRep (see, e.g., [CHK11]) since, in MinRep, we are allowed to assign multiple labels to all nodes, including those in U.

Results. First, note that checking whether $\mathsf{MaxCov}(\Gamma) < r$ or not, for any $r \geq 1$, can be done by the following algorithms.

1. It can be done[8] in $O^\star(\binom{|U|}{r}(|\Sigma_U|)^r) = O^\star((|U| \cdot |\Sigma_U|)^r)$ time: First, enumerate all $\binom{|U|}{r}$ possible subsets U' of U and all $|\Sigma_U|^{|U'|}$ possible labelings on vertices in U'. Once we fix the labeling on U', we only need polynomial time to check whether we can label other vertices so that all vertices in U' are covered.

2. It can be done in $O^\star(|\Sigma_V|^{|V|})$ time: Enumerate all $O^\star(|\Sigma_V|^{|V|})$ possible labelings σ_V on V. After σ_V is fixed, we can find labeling σ_U on U that maximizes the number of vertices covered in U in polynomial time.

ETH can be restated as that these algorithms are the best possible when $|U| = \Theta(|V|)$, $|\Sigma_U|, |\Sigma_V| = O(1)$ and Π has the projection property. Gap-ETH asserts further that this is the case even to distinguish between $\mathsf{MaxCov}(\Gamma) = |U|$ and $\mathsf{MaxCov}(\Gamma) \leq (1 - \varepsilon)|U|$.

[8] Recall that we use $O^\star(\cdot)$ to hide factors polynomial in the input size.

Theorem 4.1. *Gap-ETH (Conjecture 2.5) is equivalent to the following statement. There exist constants $\varepsilon, \delta > 0$ such that no algorithm can take a label cover instance Γ and can distinguish between the following cases in $O(2^{\delta|U|})$ time:*

- $\mathsf{MaxCov}(\Gamma) = |U|$, *and*
- $\mathsf{MaxCov}(\Gamma) < (1-\varepsilon)|U|$.

This holds even when $|\Sigma_U|, |\Sigma_V| = O(1)$, $|U| = \Theta(|V|)$ and Π has the projection property.

The proof of Theorem 4.1 is standard. To avoid distracting the readers, we provide the sketch of the proof in Appendix E.

We will show that Theorem 4.1 can be extended to several cases, which will be useful later. First, consider when the first ($O^\star((|U| \cdot |\Sigma_U|)^r)$-time) algorithm is faster than the second one. We show that, in this case, the first algorithm is essentially the best even for $r = O(1)$, and this holds even when we know that $\mathsf{MaxCov}(\Gamma) = |U|$.

For convenience, in the statements of Theorems 4.2 to 4.4 below, we will use the notation $|\Gamma|$ to denote the size of the label cover instance; in particular, $|\Gamma| = |\Sigma_U| + |\Sigma_V| + |U| + |V|$. Furthermore, recall that the notation $O_{k,r}(\cdot)$ denotes any multiplicative factor that depends only on k and r.

Theorem 4.2 (MaxCov with Small $|U|$)**.** *Assuming Gap-ETH, there exist constants $\delta, \rho > 0$ such that, for any positive integers $k \geq r \geq \rho$, no algorithm can take a label cover instance Γ with $|U| = k$ and distinguish between the following cases in $O_{k,r}(|\Gamma|^{\delta r})$ time:*

- $\mathsf{MaxCov}(\Gamma) = k$ *and*
- $\mathsf{MaxCov}(\Gamma) < r$.

This holds even when $|\Sigma_V| = O(1)$ and Π has the projection property.

We emphasize that it is important for applications in later sections that $r = O(1)$. In fact, the main challenge in proving the theorem above is to prove it is true for r that is arbitrarily small compared to $|U|$.

Secondly, consider when the second ($O^\star(|\Sigma_V|^{|V|})$-time) algorithm is faster, i.e., when $|V| \ll |U|$. In this case, we cannot make the soundness (i.e., the parameter r in Theorem 4.2) to be arbitrarily small. (Roughly speaking, the first algorithm can become faster otherwise.) Instead, we will show that the second algorithm is essentially the best possible for soundness as small as $\gamma|U|$, for any constant $\gamma > 0$. More importantly, this holds for $|V| = O(1)$ (thus independent from the input size). This is the key property of this theorem that we will need later.

Theorem 4.3 (MaxCov with Small $|V|$)**.** *Assuming Gap-ETH, there exist constants $\delta, \rho > 0$ such that, for any positive integer $q \geq \rho$ and any $1 \geq \gamma > 0$, no algorithm can take a label cover instance Γ with $|V| = q$ and distinguish between the following cases in $O_{q,\gamma}(|\Gamma|^{\delta q})$ time:*

- $\mathsf{MaxCov}(\Gamma) = |U|$ *and*
- $\mathsf{MaxCov}(\Gamma) < \gamma|U|$.

This holds even when $|\Sigma_U| \leq (1/\gamma)^{O(1)}$.

We remark that the above label cover instance does not have the projection property.

In our final result, we turn to computing $\mathsf{MinLab}(\Gamma)$. Since $\mathsf{MaxCov}(\Gamma) = |U|$ if and only if $\mathsf{MinLab}(\Gamma) = |V|$, a statement similar to Theorem 4.1 intuitively holds for distinguishing between $\mathsf{MinLab}(\Gamma) \leq |V|$ and $\mathsf{MinLab}(\Gamma) > (1+\varepsilon)|V|$, i.e., we need $O^\star(|\Sigma_V|^{|V|})$ time. In the following theorem, we show that this gap can be substantially amplified while maintaining the property that $|V| = O(1)$ (thus independent from the input size).

Theorem 4.4 (MinLab Hardness)**.** *Assuming Gap-ETH, there exist constants $\delta, \rho > 0$ such that, for any positive integers $r \geq q \geq \rho$, no algorithm can take a label cover instance Γ with $|V| = q$ and distinguish between the following cases in $O_{q,r}(|\Gamma|^{\delta q})$ time:*

- $\mathsf{MinLab}(\Gamma) = q$ *and*
- $\mathsf{MinLab}(\Gamma) > r$.

This holds even when $|\Sigma_U| = (r/q)^{O(q)}$.

The rest of this section is devoted to proving Theorems 4.2 to 4.4.

4.2 Proof of Theorem 4.2

The proof proceeds by **compressing the left vertex set** U of a label cover instance from Theorem 4.1. More specifically, each new left vertex will be a subset of left vertices in the original instance. In the construction below, these subsets will just be random subsets of the original vertex set of a certain size; however, the only property of random subsets we will need is that they form a *disperser*. To clarify our proof, let us start by stating the definition of dispersers here. Note that, even though dispersers are often described in terms of graphs or distributions in literatures (see, e.g., [Vad12]), it is more convenient for us to describe it in terms of subsets.

Definition 4.5. *For any positive integers $m, k, \ell, r \in \mathbb{N}$ and any constant $\varepsilon \in (0,1)$, an $(m, k, \ell, r, \varepsilon)$-*disperser *is a collection $\mathcal{I}$ of k subsets $I_1, \ldots, I_k \subseteq [m]$ each of size ℓ such that the union of any r different subsets from the collection has size at least $(1-\varepsilon)m$. In other words, for any $1 \leq i_1 < \cdots < i_r \leq k$, we have $|I_{i_1} \cup \cdots \cup I_{i_r}| \geq (1-\varepsilon)m$.*

The idea of using dispersers to amplify gap in hardness of approximation bears a strong resemblance to the classical randomized graph product technique [BS92]. Indeed, similar approaches have been used before, both implicitly (e.g., [BGS98]) and explicitly (e.g., [Zuc96b; Zuc96a; Zuc07]). In fact, even the reduction we use below has been studied before by Zuckerman [Zuc96b; Zuc96a]!

What differentiates our proof from previous works is the setting of parameters. Since the reduction size (specifically, the left alphabet size $|\Sigma_U|$) blows up exponentially in ℓ and previous results aim to prove NP-hardness of approximating Clique, ℓ are chosen to be small (i.e., $O(\log m)$). On the other hand, we will choose our ℓ to be $\Theta_\varepsilon(m/r)$ since we would like to only prove a running time lower bound of the form $|\Sigma_U|^{\Omega(r)}$. Interestingly, dispersers for our regime of parameters are easier to construct deterministically and we will sketch the construction in Subsection 4.2.1. Note that this construction immediately implies derandomization of our reduction.

The exact dependency of parameters can be found in the claim below, which also states that random subsets will be a disperser for such choice of parameters with high probability. Here and throughout the proof, k and r should be thought of as constants where $k \gg r$; these are the same k, r as the ones in the statement of Theorem 4.2.

Claim 4.6. *For any positive integers $m, k, r \in \mathbb{N}$ and any constant $\varepsilon \in (0,1)$, let $\ell = \max\{m, \lceil 3m/(\varepsilon r)\rceil\}$ and let $I_1, \ldots, I_k$ be ℓ-element subsets of $[m]$ drawn uniformly independently at random. If $\ln k \leq m/r$, then $\mathcal{I} = \{I_1, \ldots, I_k\}$ is an $(m, k, \ell, r, \varepsilon)$-disperser with probability at least $1 - e^{-m}$.*

Proof. When $\ell = m$, the statement is obviously true; thus, we assume w.l.o.g. that $\ell = \lceil 3m/(\varepsilon r)\rceil$. Consider any indices $i_1, \ldots, i_r$ such that $1 \leq i_1 < \cdots < i_r \leq k$. We will first compute the probability that $|I_{i_1} \cup \cdots \cup I_{i_r}| < (1-\varepsilon)m$ and then take the union bound over all such $(i_1, \ldots, i_r)$'s.

Observe that $|I_{i_1} \cup \cdots \cup I_{i_r}| < (1-\varepsilon)m$ if and only if there exists a set $S \subseteq [m]$ of size less than $(1-\varepsilon)m$ such that $I_{i_1}, \ldots, I_{i_r} \subseteq S$. For a fixed set $S \subseteq [m]$ of size less than $(1-\varepsilon)m$, since $I_{i_1}, \ldots, I_{i_r}$ are independently drawn random ℓ-element subsets of $[m]$, we have

$$\Pr[I_{i_1}, \ldots, I_{i_r} \subseteq S] = \prod_{j \in [r]} \Pr[I_j \subseteq S] = \left(\frac{\binom{|S|}{\ell}}{\binom{m}{\ell}}\right)^r \leq \left(\frac{|S|}{m}\right)^{\ell r} < (1-\varepsilon)^{\ell r} \leq e^{-\varepsilon \ell r} < e^{-3m}.$$

Taking the union bound over all such S's, we have

$$\Pr[|I_{i_1} \cup \cdots \cup I_{i_r}| < (1-\varepsilon)m] < \sum_{S \subseteq [m], |S| < (1-\varepsilon)m} e^{-3m} < 2^m \cdot e^{-3m} < e^{-2m}.$$

Finally, taking the union bound over all $(i_1, \ldots, i_r)$'s gives us the desired probabilistic bound:

$$\Pr[\mathcal{I} \text{ is not an } (m, k, \ell, r, \varepsilon)\text{-disperser}] \leq \sum_{1 \leq i_1 < \cdots < i_r \leq k} e^{-2m} \leq k^r \cdot e^{-2m} < e^{-m},$$

where the last inequality comes from our assumption that $\ln k \leq m/r$. □

With the definition of dispersers and the above claim ready, we move on to prove Theorem 4.2.

Proof of Theorem 4.2. First, we take a label cover instance $\widetilde{\Gamma} = (\widetilde{G} = (\widetilde{U}, \widetilde{V}, \widetilde{E}), \Sigma_{\widetilde{U}}, \Sigma_{\widetilde{V}}, \widetilde{\Pi})$ as in Theorem 4.1. We may assume that $|\Sigma_{\widetilde{U}}|, |\Sigma_{\widetilde{V}}| = O(1)$, and $|\widetilde{U}| = \Theta(|\widetilde{V}|)$. Moreover, let $m = |\widetilde{U}|$ and $n = |\widetilde{V}|$; for convenience, we rename the vertices in $\widetilde{U}$ and $\widetilde{V}$ so that $\widetilde{U} = [m]$ and $\widetilde{V} = [n]$. Note that it might be useful for the readers to think of $\widetilde{\Gamma}$ as a 3-SAT instance where $\widetilde{U}$ is the set of clauses and $\widetilde{V}$ is the set of variables.

We recall the parameter ε from Theorem 4.1 and the parameters k, r from the statement of Theorem 4.2. We introduce a new parameter $\ell = 3m/(\varepsilon r)$ and assume w.l.o.g. that ℓ is an integer.

The new label cover (MaxCov) instance $\Gamma = (G = (U, V, E), \Sigma_U, \Sigma_V, \Pi)$ is defined as follows.

- The right vertices and right alphabet set remain unchanged, i.e., $V = \widetilde{V}$ and $\Sigma_V = \Sigma_{\widetilde{V}}$.
- There will be k vertices in U where each vertex is a random set of ℓ vertices of $\widetilde{U}$. More specifically, we define $U = \{I_1, \ldots, I_k\}$ where each I_i is a random ℓ-element subsets of $[m]$ drawn independently of each other.
- The left alphabet set Σ_U is $\Sigma_{\widetilde{U}}^{\ell}$. For each $I \in U$, we view each label $\alpha \in \Sigma_U$ as a tuple $(\alpha_u)_{u \in I} \in (\Sigma_{\widetilde{U}})^I$; this is a partial assignment to all vertices $u \in I$ in the original instance $\widetilde{\Gamma}$.
- We create an edge between $I \in U$ and $v \in V$ in E if and only if there exists $u \in I$ such that $uv \in \widetilde{E}$. More formally, $E = \{Iv : I \cap N_{\widetilde{G}}(v) \neq \emptyset\}$.

- Finally, we define the constraint Π_{Iv} for each $Iv \in E$. As stated above, we view each $\alpha \in \Sigma_U$ as a partial assignment $(\alpha_u)_{u \in I}$ for $I \subseteq \widetilde{U}$. The constraint Π_{Iv} then contains all (α, β) such that (α_u, β) satisfies the constraint $\widetilde{\Pi}_{uv}$ for every $u \in I$ that has an edge to v in $\widetilde{\Gamma}$. More precisely, $\Pi_{Iv} = \{(\alpha, \beta) = ((\alpha_u)_{u \in I}, \beta) : \forall u \in I \cap N_{\widetilde{G}}(v), (\alpha_u, \beta) \in \widetilde{\Pi}_{uv}\}$.

Readers who prefer the 3-SAT/CSP viewpoint of label cover may think of each I_i as a collection of clauses in the 3-SAT instance that are joined by an operator **AND**, i.e., the assignment must satisfy all the clauses in I_i simultaneously in order to satisfy I_i.

We remark that, if $\widetilde{\Pi}$ has the projection property, then Π also has the projection property.

Completeness. Suppose there is a labeling $(\sigma_{\widetilde{U}}, \sigma_{\widetilde{V}})$ of $\widetilde{\Gamma}$ that covers all $|\widetilde{U}|$ left vertices. We take $\sigma_V = \sigma_{\widetilde{V}}$ and construct σ_U by setting $\sigma_U(I) = (\sigma_{\widetilde{U}}(u))_{u \in I}$ for each $I \in U$. Since $(\sigma_{\widetilde{U}}, \sigma_{\widetilde{V}})$ covers all the vertices of $\widetilde{U}$, (σ_U, σ_V) also covers all the vertices of U. Therefore, $\mathsf{MaxCov}(\Gamma) = |U|$.

Soundness. To analyze the soundness of the reduction, first recall Claim 4.6 that $\{I_1, \ldots, I_k\}$ is an $(m, k, \ell, r, \varepsilon)$-disperser with high probability. Conditioned on this event happening, we will prove the soundness property, i.e., that if $\mathsf{MaxCov}(\widetilde{\Gamma}) < (1-\varepsilon)|\widetilde{U}|$, then $\mathsf{MaxCov}(\Gamma) < r$.

We will prove this by contrapositive. Assume that there is a labeling (σ_U, σ_V) that covers at least r left vertices $I_{i_1}, \cdots, I_{i_r} \in U$. We construct a labeling $(\sigma_{\widetilde{U}}, \sigma_{\widetilde{V}})$ as follows. First, $\sigma_{\widetilde{V}}$ is simply set to σ_V. Moreover, for each $u \in I_{i_1} \cup \cdots \cup I_{i_r}$, let $\sigma_{\widetilde{U}}(u) = (\sigma_U(I_{i_j}))_u$ where $j \in [r]$ is an index such that $u \in I_{i_j}$; if there are multiple such j's, then we may pick an arbitrary one. Finally, for $u \in U \setminus (I_{i_1} \cup \cdots \cup I_{i_r})$, we set $\sigma_{\widetilde{U}}(u)$ arbitrarily.

We claim that every $u \in I_{i_1} \cup \cdots \cup I_{i_r}$ is covered by $(\sigma_{\widetilde{U}}, \sigma_{\widetilde{V}})$ in the original instance $\widetilde{\Gamma}$. To see that this is the case, recall that $\sigma_{\widetilde{U}}(u) = (\sigma_U(I_{i_j}))_u$ for some $j \in [r]$ such that $u \in I_{i_j}$. For every $v \in V$, if $uv \in E$, then, from how the constraint $\Pi_{I_{i_j} v}$ is defined, we have $(\sigma_{\widetilde{U}}(u), \sigma_{\widetilde{V}}(v)) = (\sigma_U(I_{i_j})_u, \sigma_V(v)) \in \widetilde{\Pi}_{uv}$. In other words, u is indeed covered by $(\sigma_{\widetilde{U}}, \sigma_{\widetilde{V}})$.

Hence, $(\sigma_{\widetilde{U}}, \sigma_{\widetilde{V}})$ covers at least $|I_{i_1} \cup \cdots \cup I_{i_r}| \geq (1-\varepsilon)m$, where the inequality comes from the definition of dispersers. As a result, $\mathsf{MaxCov}(\widetilde{\Gamma}) \geq (1-\varepsilon)|\widetilde{U}|$, completing the soundness proof.

Running Time Lower Bound. Our construction gives a MaxCov instance Γ with $|U| = k$ and $|\Sigma_U| = |\Sigma_{\widetilde{U}}|^\ell = 2^{\Theta(m/(\varepsilon r))}$, whereas $|V|$ and $|\Sigma_V|$ remain n and $O(1)$, respectively. Assume that Gap-ETH holds and let δ_0 be the constant in the running time lower bound in Theorem 4.1. Let δ be any constant such that $0 < \delta < \delta_0 \varepsilon / c$ where c is the constant such that $|\Sigma_U| \leq 2^{cm/(\varepsilon r)}$.

Suppose for the sake of contradiction that, for some $k \geq r \geq \rho$, there is an algorithm that distinguishes whether $\mathsf{MaxCov}(\Gamma) = k$ or $\mathsf{MaxCov}(\Gamma) < r$ in $O_{k,r}(|\Gamma|^{\delta r})$ time. Observe that, in our reduction, $|U|, |V|, |\Sigma_V| = |\Sigma_U|^{o(1)}$. Hence, the running time of the algorithm on input Γ is at most $O_{k,r}(|\Sigma_U|^{\delta r(1+o(1))}) \leq O_{k,r}(|\Sigma_U|^{\delta_0 \varepsilon r / c}) \leq O(2^{\delta_0 m})$ where the first inequality comes from our choice of δ and the second comes from $|\Sigma_U| \leq 2^{cm/(\varepsilon r)}$. Thanks to the completeness and soundness of the reduction, this algorithm can also distinguish whether $\mathsf{MaxCov}(\widetilde{\Gamma}) = |\widetilde{U}|$ or $\mathsf{MaxCov}(\widetilde{\Gamma}) < (1-\varepsilon)|\widetilde{U}|$ in time $O(2^{\delta_0 m})$. From Theorem 4.1, this is indeed a contradiction. □

4.2.1 Derandomization

While the reduction in the proof of Theorem 4.2 is a randomized reduction, it can be derandomized quite easily. We sketch the ideas behind the derandomization below.

Notice that the only property we need from the random ℓ-element subsets $I_1, \ldots, I_k$ is that it forms an $(m, k, \ell, r, \varepsilon)$-disperser. Hence, to derandomize the reduction, it suffices to deterministically construct such a disperser in $2^{o(n)}$ time.

To do so, let us first note that Lemma 4.6 implies that an $(m', k, \ell', r, \varepsilon)$-disperser exists where $m' = r \ln k$ and $\ell' = 3m'/(\varepsilon r)$. For convenience, we assume w.l.o.g. that m', ℓ' are integers and that m' divides m. Since m' is now small, we can find such a disperser by just enumerating over every possible collection of k subsets of $[m']$ each of size ℓ' and checking whether it has the desired property; this takes only $(2^{m'})^k (k)^r \operatorname{poly}(m') = 2^{O(rk \log k)}$ time, which is acceptable for us since r and k are both constants. Let the $(m', k, \ell', r, \varepsilon)$-disperser that we find be $\{I'_1, \ldots, I'_k\}$. Finally, to get from here to the intended $(m, k, \ell, r, \varepsilon)$-disperser, we only need to view $[m]$ as $[m/m'] \times [m']$ and let $I_1 = [m/m'] \times I'_1, \ldots, I_k = [m/m'] \times I'_k$. It is not hard to check that $\{I_1, \ldots, I_k\}$ is indeed an $(m, k, \ell, r, \varepsilon)$-disperser, which concludes our sketch.

4.3 Proof of Theorem 4.3

The proof proceeds by **compressing the right vertex set** V of a label cover instance from Theorem 4.1 plus amplifying the hardness gap. The gap amplification step is similar to that in the proof of Theorem 4.2 except that, since here $\mathsf{MaxCov}(\Gamma)$ is not required to be constant in the soundness case, we can simply take all subsets of appropriate sizes instead of random subsets as in the previous proof; this also means that our reduction is deterministic and thus requires no derandomization.

Proof of Theorem 4.3. First, we take a label cover instance $\widetilde{\Gamma} = (\widetilde{G} = (\widetilde{U}, \widetilde{V}, \widetilde{E}), \Sigma_{\widetilde{U}}, \Sigma_{\widetilde{V}}, \widetilde{\Pi})$ as in Theorem 4.1. We may assume that $|\Sigma_{\widetilde{U}}|, |\Sigma_{\widetilde{V}}| = O(1)$, and $|\widetilde{U}| = \Theta(|\widetilde{V}|)$. For convenience, we assume w.l.o.g. that $\widetilde{U} = [m]$ and $\widetilde{V} = [n]$. Again, it might be useful for the readers to think of $\widetilde{\Gamma}$ as a 3-SAT instance where $\widetilde{U}$ are the set of clauses and $\widetilde{V}$ are the set of variables.

Recall the parameter ε from Theorem 4.1 and the parameters q, γ from Theorem 4.3. Let $\ell = \ln(1/\gamma)/\varepsilon$. We assume w.l.o.g. that ℓ is an integer and that n is divisible by q. The new label cover (MaxCov) instance $\Gamma = (G = (U, V, E), \Sigma_U, \Sigma_V, \Pi)$ is defined as follows.

- First, we partition $\widetilde{V} = [n]$ into q parts $J_1, \ldots, J_q$, each of size n/q. We then let $V = \{J_1, \ldots, J_q\}$. In other words, we merge n/q vertices of $\widetilde{V}$ into a single vertex in V.
- Let U be $\binom{[m]}{\ell}$, the collection of all ℓ-element subsets of $[m] = \widetilde{U}$.
- The left alphabet set Σ_U is $\Sigma_{\widetilde{U}}^{\ell}$. For each $I \in U$, we view each label $\alpha \in \Sigma_U$ as a tuple $(\alpha_u)_{u \in I} \in (\Sigma_{\widetilde{U}})^I$; this is a partial assignment to all vertices $u \in I$ in the original instance $\widetilde{\Gamma}$.
- Our graph G is simply a complete bipartite graph, i.e., for every $I \in U$ and $J \in V$, $IJ \in E(G)$.
- The label set of V is $\Sigma_V = \Sigma_{\widetilde{V}}^{n/q}$, and the label set of U is $\Sigma_U = \Sigma_{\widetilde{U}}^{\ell}$. For each $I \in U$, we view each label $\alpha \in \Sigma_U$ as a tuple $(\alpha_u)_{u \in I} \in (\Sigma_{\widetilde{U}})^I$; this is simply a partial assignment to all vertices $u \in I$ in the original instance $\widetilde{\Gamma}$. Similarly, for each $J \in V$, we view each label $\beta \in \Sigma_V$ as $(\beta_v)_{v \in J} \in (\Sigma_{\widetilde{V}})^J$.
- Finally, we define Π_{IJ} for each $IJ \in E$. The constraint Π_{IJ} contains all (α, β) such that (α_u, β_v) satisfies the constraint $\widetilde{\Pi}_{uv}$ for every $u \in I, v \in J$ such that $uv \in \widetilde{E}$. More precisely, $\Pi_{IJ} = \{(\alpha, \beta) = ((\alpha_u)_{u \in I}, (\beta_v)_{v \in J}) : \forall u \in I, v \in J \text{ such that } uv \in \widetilde{E}, (\alpha_u, \beta_v) \in \widetilde{\Pi}_{uv}\}$.

We remark that Π may not have the projection property even when $\widetilde{\Pi}$ has the property.

Completeness. Suppose that there is a labeling $(\sigma_{\widetilde{U}}, \sigma_{\widetilde{V}})$ of $\widetilde{\Gamma}$ that covers all $|\widetilde{U}|$ left vertices. We construct (σ_U, σ_V) by setting $\sigma_U(I) = (\sigma_{\widetilde{U}}(u))_{u \in I}$ for each $I \in U$ and $\sigma_V(J) = (\sigma_{\widetilde{V}}(v))_{v \in J}$ for each $J \in V$. It is easy to see that (σ_U, σ_V) covers all the vertices of U. Therefore, $\mathsf{MaxCov}(\Gamma) = |U|$.

Soundness. Suppose that $\mathsf{MaxCov}(\widetilde{\Gamma}) < (1-\varepsilon)|\widetilde{U}|$. Consider any labeling (σ_U, σ_V) of Γ; we will show that (σ_U, σ_V) covers less than $\gamma|U|$ left vertices.

Let $I_1, \ldots, I_t \in U$ be the vertices covered by (σ_U, σ_V). Analogous to the proof of Theorem 4.2, we define a labeling $(\sigma_{\widetilde{U}}, \sigma_{\widetilde{V}})$ as follows. First, $\sigma_{\widetilde{V}}$ is naturally defined from σ_V by $\sigma_{\widetilde{V}} = \sigma_V(J)_v$ where J is the partition that contains v. Moreover, for each $u \in I_{i_1} \cup \cdots \cup I_{i_r}$, let $\sigma_{\widetilde{U}}(u) = (\sigma_U(I_{i_j}))_u$ where $j \in [r]$ is an index such that $u \in I_{i_j}$; for $u \in U \setminus (I_{i_1} \cup \cdots \cup I_{i_r})$, we set $\sigma_{\widetilde{U}}(u)$ arbitrarily.

Similar to the proof of Theorem 4.2, it is not hard to see that every vertex in $I_1 \cup \cdots \cup I_t$ is covered by $(\sigma_{\widetilde{U}}, \sigma_{\widetilde{V}})$ in $\widetilde{\Gamma}$. Since $\mathsf{MaxCov}(\widetilde{\Gamma}) < (1-\varepsilon)|\widetilde{U}|$, we can conclude that $|I_1 \cup \cdots \cup I_t| < (1-\varepsilon)|\widetilde{U}|$. Since each I_i is simply an ℓ-size subset of $I_1 \cup \cdots \cup I_t$, we can conclude that

$$t < \binom{(1-\varepsilon)|\widetilde{U}|}{\ell} \leq (1-\varepsilon)^\ell \binom{|\widetilde{U}|}{\ell} = (1-\varepsilon)^\ell |U| \leq e^{-\varepsilon\ell}|U| = \gamma|U|.$$

Hence, (σ_U, σ_V) covers less than $\gamma|U|$ left vertices as desired.

Running Time Lower Bound. Our construction gives a MaxCov instance Γ with $|V| = q$ and $|\Sigma_V| = |\Sigma_{\widetilde{V}}|^{n/q} = 2^{\Theta(n/q)}$; note also that $|U| = m^\ell$ and $|\Sigma_U| = |\Sigma_{\widetilde{U}}|^\ell = (1/\gamma)^{O(1)}$. Assume that Gap-ETH holds and let δ_0 be the constant from Theorem 4.1. Moreover, let δ be any positive constant such that $\delta < \delta_0/c$ where c is the constant such that $|\Sigma_V| \leq 2^{cm/q}$.

Suppose for the sake of contradiction that, for some $q \geq \rho$ and $1 \geq \gamma > 0$, there is an algorithm that distinguishes whether $\mathsf{MaxCov}(\Gamma) = |U|$ or $\mathsf{MaxCov}(\Gamma) < \gamma|U|$ in $O_{q,\gamma}(|\Gamma|^{\delta q})$ time. Observe that, in our reduction, $|U|, |V|, |\Sigma_U| = |\Sigma_V|^{o(1)}$. Hence, the running time of the algorithm on input Γ is $O_{q,\gamma}(|\Sigma_V|^{\delta q(1+o(1))}) \leq O_{q,\gamma}(|\Sigma_V|^{\delta_0 q/c}) \leq O(2^{\delta_0 m})$ where the first inequality comes from our choice of δ and the second comes from $|\Sigma_V| \leq 2^{cm/q}$. Thanks to the completeness and soundness of the reduction, this algorithm can also distinguish whether $\mathsf{MaxCov}(\widetilde{\Gamma}) = |\widetilde{U}|$ or $\mathsf{MaxCov}(\widetilde{\Gamma}) < (1-\varepsilon)|\widetilde{U}|$ in time $O(2^{\delta_0 m})$. From Theorem 4.1, this is a contradiction. □

4.4 Proof of Theorem 4.4

We conclude this section with the proof of Theorem 4.4. The proof proceeds simply by showing that if an algorithm can distinguish between the two cases in the statement of Theorem 4.4, it can also distinguish between the two cases in Theorem 4.3 (with an appropriate value of γ).

Proof of Theorem 4.4. Consider the label cover instance $\Gamma = (G = (U, V, E), \Sigma_U, \Sigma_V, \Pi)$ given by Theorem 4.3 when $\gamma = (r/q)^{-q}$. Let us assume w.l.o.g. that there is no isolated vertex in G.

Completeness. If $\mathsf{MaxCov}(\Gamma) = |U|$, then there is a labeling $\sigma_U : U \to \Sigma_U$ and $\sigma_V : V \to \Sigma_V$ that covers every edge; this also induces a multi-labeling that covers every edge. Hence, $\mathsf{MinLab}(\Gamma) = |V|$.

Soundness. We will prove by contrapositive. Suppose that $\mathsf{MinLab}(\Gamma) \leq r$. This implies that there exists a multi-labeling $\sigma_U : U \to \Sigma_U$ and $\sigma_V : V \to 2^{\Sigma_V}$ such that $\sum_{v \in V} |\sigma_V(v)| \leq r$ and every vertex is covered. Since there is no isolated vertex in G, $\sigma_V(v) \neq \emptyset$ for all $v \in V$.

Consider $\sigma_V^{\mathrm{rand}} : V \to \Sigma_V$ sampled randomly by, for each $v \in V$, independently pick a random element of $\sigma_V(v)$ and let $\sigma_V^{\mathrm{rand}}(v)$ be this element. Let us consider the expected number of $u \in U$

that are covered by the labeling $(\sigma_U, \sigma_V^{\text{rand}})$. From linearity of expectation, we can write this as

$$\begin{aligned}
\mathop{\mathbb{E}}_{\sigma_V^{\text{rand}}} |\{u \in U \mid (\sigma_U, \sigma_V^{\text{rand}}) \text{ covers } u\}| &= \sum_{u \in U} \Pr_{\sigma_V^{\text{rand}}} \left[(\sigma_U, \sigma_V^{\text{rand}}) \text{ covers } u \right] \\
&= \sum_{u \in U} \prod_{v \in N(u)} \Pr \left[(\sigma_U(u), \sigma_V^{\text{rand}}(v)) \in \Pi_{uv} \right] \\
&\geq \sum_{u \in U} \prod_{v \in N(u)} |\sigma_V(v)|^{-1} \\
&\geq \sum_{u \in U} \prod_{v \in V} |\sigma_V(v)|^{-1} \\
\text{(From AM-GM inequality)} &\geq \sum_{u \in U} \left(\frac{1}{q} \sum_{v \in V} |\sigma_V(v)| \right)^{-q} \\
&\geq \sum_{u \in U} (r/q)^{-q} \\
&= \gamma |U|.
\end{aligned}$$

where the first inequality comes from the fact that there exists $\beta \in \sigma_V(v)$ such that $(\sigma_U(u), \beta) \in \Pi_{uv}$. This implies that $\mathsf{MaxCov}(\Gamma) \geq \gamma |U|$, which concludes our proof. □

5 Hardness for Combinatorial Problems

5.1 Maximum Clique

Recall that, for any graph G, $\mathsf{Clique}(G)$ denotes the maximum size of any clique in G. Observe that we can check if there is a clique of size r by checking if any subset of r vertices forms a clique, and there are $\binom{|V(G)|}{r} = O(|V(G)|^r)$ possible such subsets. We show that this is essentially the best we can do even when we are given a promise that a clique of size $q \gg r$ exists:

Theorem 5.1. *Assuming Gap-ETH, there exist constants $\delta, r_0 > 0$ such that, for any positive integers $q \geq r \geq r_0$, no algorithm can take a graph G and distinguish between the following cases in $O_{q,r}(|V(G)|^{\delta r})$ time:*

- $\mathsf{Clique}(G) \geq q$ *and*
- $\mathsf{Clique}(G) < r$.

The above theorem simply follows from plugging the FGLSS reduction below to Theorem 4.2.

Theorem 5.2 ([FGLSS96])**.** *Given a label cover instance $\Gamma = (G = (U, V, E), \Sigma_U, \Sigma_V, \Pi)$ with projection property as in Section 4, there is a reduction that produces a graph H_Γ such that $|V(H_\Gamma)| = |U||\Sigma_U|$ and $\mathsf{Clique}(H_\Gamma) = \mathsf{MaxCov}(\Gamma)$. The reduction takes $O(|V(H_\Gamma))|^2 |V|)$ time.*

For clarity, we would like to note that, while the original graph defined in [FGLSS96] is for multi-prover interactive proof, analogous graphs can be constructed for CSPs and label cover instances as well. In particular, in our case, the graph can be defined as follows:

- The vertex set $V(H_\Gamma)$ is simply $U \times \Sigma_U$.

- There is an edge between two vertices $(u, \alpha), (u', \alpha') \in V(H_\Gamma)$ if and only if $\Pi_{uv}(\alpha) = \Pi_{u'v}(\alpha')$ (i.e., recall that we have a projection constraint, so we can represent the constraint Π_{uv} as a function $\Pi_{uv} : \Sigma_U \to \Sigma_V$.)

Proof of Theorem 5.1. Assume that Gap-ETH holds and let δ, ρ be the constants from Theorem 4.2. Let $r_0 = \max\{\rho, 2/\delta\}$. Suppose for the sake of contradiction that, for some $q \geq r \geq r_0$, there is an algorithm $\mathbb{A}$ that distinguishes between $\mathsf{Clique}(G) \geq q$ and $\mathsf{Clique}(G) < r$ in $O_{q,r}(|V(G)|^{\delta r})$ time.

Given a label cover instance Γ with projection property, we can use $\mathbb{A}$ to distinguish whether $\mathsf{MaxCov}(\Gamma) \geq q$ or $\mathsf{MaxCov}(\Gamma) < r$ as follows. First, we run the FGLSS reduction to produce a graph H_Γ, and we then use $\mathbb{A}$ to decide whether $\mathsf{Clique}(H_\Gamma) \geq q$ or $\mathsf{Clique}(H_\Gamma) < r$. From $\mathsf{Clique}(H_\Gamma) = \mathsf{MaxCov}(\Gamma)$, this indeed correctly distinguishes between $\mathsf{MaxCov}(\Gamma) \geq q$ and $\mathsf{MaxCov}(\Gamma) < r$; moreover, the running time of the algorithm is $O_{q,r}(|V(H_\Gamma)|^{\delta r}) + O(|V(H_\Gamma))|^2|V|) \leq O_{q,r}(|\Gamma|^{\delta r})$ where the term $O(|V(H_\Gamma))|^2|V|)$ comes from the running time used to produce H_Γ. From Theorem 4.2, this is a contradiction, which concludes our proof. □

As a corollary of Theorem 5.1, we immediately arrive at FPT inapproximability of Clique and MIS.

Corollary 5.3 (Clique is inherently enumerative)**.** *Assuming Gap-ETH, the Maximum Clique and Maximum Independent Set problems are inherently enumerative and thus FPT inapproximable.*

5.2 Set Cover, Dominating Set, and Hitting Set

For convenience, we will be working with the *Set Cover* problem, which is computationally equivalent to DomSet (see Appendix D).

Let $\mathcal{U}$ be a ground set (or a universe). A set system $\mathcal{S}$ over $\mathcal{U}$ is a collection of subsets $\mathcal{S} = \{S_1, \ldots, S_m\}$ where $S_i \subseteq \mathcal{U}$ for all $i \in [m]$. We say that $\mathcal{S}' \subseteq \mathcal{S}$ is a feasible set cover of $(\mathcal{U}, \mathcal{S})$ if $\bigcup_{X \in \mathcal{S}'} X = \mathcal{U}$. In the Set Cover problem (SetCov), we are given a set system $(\mathcal{U}, \mathcal{S})$ and we are interested in finding a set cover $\mathcal{S}'$ with minimum cardinality $|\mathcal{S}'|$. Let $\mathsf{SetCov}(\mathcal{U}, \mathcal{S})$ denote the value of the optimal set cover for $(\mathcal{U}, \mathcal{S})$. SetCov has an alternative formulation called *Hitting Set* (HitSet) where the goal is to find a collection of elements with minimum cardinality that *hits* every subset, i.e., each subset must contain at least one chosen element. It is not hard to see that one may interchange the role of elements and subsets to get an instance of HitSet from that of SetCov and vice versa.

Note that for any set cover instance $(\mathcal{U}, \mathcal{S})$, checking whether there is a set cover of size at most q can be done in $O^\star(|\mathcal{S}|^q)$ time by enumerating all $\binom{|\mathcal{S}|}{q}$ subsets of $\mathcal{S}$ of size q. We show that this is more or less the best we can do: Even when the algorithm is promised the existence of a set cover of size q (for some constant q), it cannot find a set cover of size $f(q)$ for any computable function f in time $O_q(|\mathcal{S}||\mathcal{U}|)^{\delta q}$ for some constant $\delta > 0$ independent of q and f.

5.2.1 Results

Our main technical contribution in this section is summarized in the following theorem:

Theorem 5.4. *There is a reduction that on input $\Gamma = (G = (U, V, E), \Sigma_U, \Sigma_V, \Pi)$ of* MinLab *instance, produces a set cover instance $(\mathcal{U}, \mathcal{S})$ such that*

- $\mathsf{MinLab}(\Gamma) = \mathsf{SetCov}(\mathcal{U}, \mathcal{S})$

- $|\mathcal{U}| = |U||V|^{|\Sigma_U|}$ *and* $|\mathcal{S}| = |V||\Sigma_V|$
- *The reductions runs in time* $poly(|\mathcal{U}|, |\mathcal{S}|)$

We defer the proof of this theorem to Section 5.2.2. For now, let us demonstrate that, by combining Theorem 5.4 and Theorem 4.3, we can derive hardness of approximating SetCov:

Theorem 5.5. *Assuming Gap-ETH, there exist universal constants* $\delta, q_0 > 0$ *such that, for any positive integers* $r \geq q \geq q_0$*, no algorithm can take a set cover instance* $(\mathcal{U}, \mathcal{S})$*, and distinguish between the following cases in* $O_{q,r}((|\mathcal{S}||\mathcal{U}|)^{\delta q})$ *time:*

- $\mathsf{SetCov}(\mathcal{U}, \mathcal{S}) \leq q$.
- $\mathsf{SetCov}(\mathcal{U}, \mathcal{S}) > r$.

Proof. Assume that Gap-ETH holds and let δ, ρ be the constants from Theorem 4.4. Let $q_0 = \max\{\rho, c/\delta\}$ where c is the constant such that the running time of the reduction in Theorem 5.4 is $O((|\mathcal{U}||\mathcal{S}|)^c)$. Suppose for the sake of contradiction that, for some $r \geq q \geq q_0$, there is an algorithm $\mathbb{A}$ that distinguishes between $\mathsf{SetCov}(\mathcal{U}, \mathcal{S}) \leq q$ and $\mathsf{SetCov}(\mathcal{U}, \mathcal{S}) > r$ in $O_{q,r}((|\mathcal{S}||\mathcal{U}|)^{\delta q})$ time.

Given a label cover instance Γ where $|V|, |\Sigma_U| = O_{q,r}(1)$, we can use $\mathbb{A}$ to distinguish whether $\mathsf{MinLab}(\Gamma) \leq q$ or $\mathsf{MinLab}(\Gamma) > r$ as follows. First, we run the reduction from Theorem 5.4 to produce a SetCov instance $(\mathcal{U}, \mathcal{S})$, and we then use $\mathbb{A}$ to decide whether $\mathsf{SetCov}(\mathcal{U}, \mathcal{S}) \leq q$ or $\mathsf{SetCov}(\mathcal{U}, \mathcal{S}) > r$. From $\mathsf{SetCov}(\mathcal{U}, \mathcal{S}) = \mathsf{MinLab}(\Gamma)$, this indeed correctly distinguishes between $\mathsf{MinLab}(\Gamma) \leq q$ and $\mathsf{MinLab}(\Gamma) > r$; moreover, the running time of the algorithm is $O_{q,r}((|\mathcal{U}||\mathcal{S}|)^{\delta q}) + O((|\mathcal{U}||\mathcal{S}|)^c) \leq O_{q,r}(|\Gamma|^{\delta q})$ where the term $O((|\mathcal{U}||\mathcal{S}|)^c)$ comes from the running time used to produce $(\mathcal{U}, \mathcal{S})$. From Theorem 4.4, this is a contradiction, which concludes our proof. □

As a corollary of Theorem 5.5, we immediately arrive at FPT inapproximability of SetCov, HitSet and DomSet.

Corollary 5.6. *Assuming Gap-ETH, Set cover, Dominating set and Hitting set are inherently enumerative and thus FPT inapproximable.*

5.2.2 Proof of Theorem 5.4

Our construction is based on a standard hypercube set system, as used by Feige [Fei98] in proving the hardness of the *k-Maximum Coverage* problem. We explain it here for completeness.

Hypercube set system: Let $z, k \in \mathbb{N}$ be parameters. The hypercube set system $H(z, k)$ is a set system $(\mathcal{U}, \mathcal{S})$ with the ground set $\mathcal{U} = [z]^k$. We view each element of $\mathcal{U}$ as a length-k vector $\vec{x}$ where each coordinate assumes a value in $[z]$. There is a collection of *canonical sets* $\mathcal{S} = \{X_{i,a}\}_{i \in [z], a \in [k]}$ defined as

$$X_{i,a} = \{\vec{x} : \vec{x}_a = i\}$$

In other words, each set $X_{i,a}$ contains the vectors whose a^{th} coordinate is i. A nice property of this set system is that it can only be covered completely if all canonical sets corresponding to some a^{th} coordinate are chosen.

Proposition 5.7. *Consider any sub-collection* $\mathcal{S}' \subseteq \mathcal{S}$*. We have* $\bigcup \mathcal{S}' = \mathcal{U}$ *if and only if there is a value* $a \in [k]$ *for which* $X_{1,a}, X_{2,a}, \ldots, X_{z,a} \in \mathcal{S}'$.

Proof. The if part is obvious. For the "only if" part, assume that for each $a \in [k]$, there is a value $i_a \in [z]$ for which $X_{i_a,a}$ is not in $\mathcal{S}'$. Define the vector $\vec{x}$ by $\vec{x}_a = i_a$. Notice that $\vec{x}$ does not belong to any set in $\mathcal{S}'$ (By definition, if $X_{i',a'}$ contains $\vec{x}$, then it must be the case that $\vec{x}_{a'} = i' = i_{a'}$.) $\square$

The construction: Our reduction starts from the MinLab instance $\Gamma = (G, \Sigma_U, \Sigma_V, \Pi)$. We will create the set system $\mathcal{I} = (\mathcal{U}, \mathcal{S})$. We make $|U|$ different copies of the hypercube set system: For each vertex $u \in U$, we have the hypercube set system $(\mathcal{U}^u, \mathcal{S}^u) = H(N_G(u), \Sigma_U)$, i.e., the ground set $\mathcal{U}^u$ is a copy of $N_G(u)^{\Sigma_U}$ and $\mathcal{S}^u$ contains $|N_G(u)||\Sigma_U|$ "virtual" sets, that we call $\{S^u_{v,a}\}_{v \in N_G(u), a \in \Sigma_U}$ where each such set corresponds to a canonical set of the hypercube. We remark that these virtual sets are not the eligible sets in our instance $\mathcal{I}$. For each vertex $v \in V$, for each label $b \in \Sigma_V$, we define a set

$$S_{v,b} = \bigcup_{u \in N_G(v), (a,b) \in \Pi_{uv}} S^u_{v,a}$$

The set system $(\mathcal{U}, \mathcal{S})$ in our instance is simply:

$$\mathcal{U} = \bigcup_{u \in U} \mathcal{U}^u \qquad \text{and} \qquad \mathcal{S} = \{S_{v,b} : v \in V, b \in \Sigma_V\}$$

Notice that the number of sets is $|V||\Sigma_V|$, and the number of elements in the ground set is $|\mathcal{U}| = |U||V|^{|\Sigma_U|}$. This completes the description of our instance.

Analysis: We argue that the optimal value of Γ is equal to the optimal of $(\mathcal{U}, \mathcal{S})$.

First, we will show that $\mathsf{MinLab}(\Gamma) \leq \mathsf{SetCov}(\mathcal{U}, \mathcal{S})$. Let $(\sigma_U, \hat{\sigma}_V)$ be a feasible MinLab cover for Γ (recall that $\hat{\sigma}_V$ is a multi-labeling, while σ_U is a labeling.) For each $v \in V$, the SetCov solution chooses the set $S_{v,b}$ for all $b \in \hat{\sigma}_V(v)$. Denote this solution by $\mathcal{S}' \subseteq \mathcal{S}$. The total number of sets chosen is exactly $\sum_v |\hat{\sigma}(v)|$, exactly matching the cost of $\mathsf{MinLab}(\Gamma)$. We argue that this is a feasible set cover: For each u, the fact that u is covered by $(\sigma_U, \hat{\sigma}_V)$ implies that, for all $v \in N_G(u)$, there is a label $b_v \in \hat{\sigma}_V(v)$ such that $(\sigma_U(u), b_v) \in \Pi_{uv}$. Notice that $S^u_{v,\sigma_U(u)} \subseteq S_{v,b_v} \in \mathcal{S}'$ for every $v \in N_G(u)$, so we have

$$\bigcup_{S \in \mathcal{S}'} S \supseteq \bigcup_{v \in N_G(u)} S_{v,b_v} \supseteq \bigcup_{v \in N_G(u)} S^u_{v,\sigma_U(u)} = \mathcal{U}^u$$

where the last equality comes from Proposition 5.7. In other words, $\mathcal{S}'$ covers all elements in $\mathcal{U}^u$. Hence, $\mathcal{S}'$ is indeed a valid SetCov solution for $(\mathcal{U}, \mathcal{S})$.

To prove the converse, consider a collection of sets $\{S_{v,b}\}_{(v,b) \in \Lambda}$ that covers the whole universe $\mathcal{U}$. We define the (multi-)labeling $\hat{\sigma}_V : V \to 2^{\Sigma_V}$ where $\hat{\sigma}_V(v) = \{b : (v,b) \in \Lambda\}$ for each $v \in V$. Clearly, $\sum_{v \in V} |\hat{\sigma}_V(v)| = |\Lambda|$, so the cost of $\hat{\sigma}_V$ as a solution for MinLab is exactly the cost of SetCov. We verify that all left vertices $u \in U$ of Γ are covered (and along the way will define $\Sigma_U(u)$ for all $u \in U$). Consider each vertex $u \in U$. The fact that the ground elements in $\mathcal{U}^u$ are covered implies that (from Proposition 5.7) there is a label $a_u \in \Sigma_U$ where all virtual sets $\{S^u_{v,a_u}\}_{v \in N_G(u)}$ are included in the solution. Therefore, for each $v \in N_G(u)$, there must be a label $b_v \in \hat{\sigma}_V(v)$ such that $a_u b_v \in \Pi_{uv}$. We simply define $\sigma_U(u) = a_u$. Therefore, the vertex u is covered by the assignment $(\sigma_U, \hat{\sigma}_V)$.

5.3 Maximum Induced Subgraph with Hereditary Properties

In this section, we prove the hardness of maximum induced subgraphs with hereditary property. Let Π be a graph property. We say that a subset $S \subseteq V(G)$ has property Π if $G[S] \in \Pi$. Denote by $A_\Pi(G)$ the maximum cardinality of a set S that has property Π.

Khot and Raman [KR00] proved a dichotomy theorem for the problem: If Π contains all independent sets but not all cliques or if Π contains all cliques but not all independent sets, then the problem is W[1]-hard. For all other Π's, the problem is in FPT. We will show that Khot and Raman's dichotomy theorem to hold even for FPT approximation as stated more precisely below.

Theorem 5.8. *Let Π be any hereditary property.*

- *If Π contains all independent sets but not all cliques or vice versa, then computing $A_\Pi(G)$ is weakly inherently enumerative (and therefore totally FPT inapproximable).*
- *Otherwise, $A_\Pi(G)$ can be computed exactly in FPT.*

Surprisingly, the fact that there is a gap in the optimum of our starting point helps make our reduction simpler than that of Khot and Raman. For convenience, let us focus only on properties Π's which contain all independent sets but not all cliques. The other case can be proved analogously. The main technical result is summarized in the following lemma.

Theorem 5.9. *Let Π be any graph property that contains all independent sets but not all cliques. Then there is a function $g_\Pi = \omega(1)$ such that the following holds:*

- *If $\alpha(G) \geq q$, then $A_\Pi(G) \geq q$.*
- *If $A_\Pi(G) \geq r$, then $\alpha(G) \geq g_\Pi(r)$.*

Proof. Since Π contains all independent set, when $\alpha(G) \geq q$, we always have $A_\Pi(G) \geq q$.

Now, to prove the converse, let $g_\Pi(r)$ denote $\max_{H \in \Pi, |V(H)|=r} \alpha(H)$. If $A_\Pi(G) = r$, then there exists a subset $S \subseteq V(G)$ of size r that has property Π; from the definition of g_Π, $\alpha(H) \geq g_\Pi(r)$, which implies that $\alpha(G) \geq g_\Pi(r)$ as well. Hence, we are only left to show that $g_\Pi = \omega(1)$.

To show that this is the case, recall the Ramsey theorem.

Theorem 5.10 (Ramsey's Theorem)**.** *For any $s, t \geq 1$, there is an integer $R(s,t)$ s.t. every graph on $R(s,t)$ vertices contains either a s-clique or a t-independent set. Moreover, $R(s,t) \leq \binom{s+t-2}{s-1}$.*

Recall that, from our assumption of Π, there exists a fixed integer s_Π such that Π does not contain an s_Π-clique. Hence, from Ramsey's Theorem, $g_\Pi(r) \geq \max\{t \mid R(s_\Pi, t) \leq r\}$. In particular, this implies that $g_\Pi(r) \geq \Omega_{s_\Pi}(r^{1/(s_\Pi - 1)})$. Hence, $\lim_{r\infty} g_\Pi(r) = \infty$ (i.e., $g_\Pi = \omega(1)$) as desired. □

In other words, the identical transformation $G \mapsto G$ is a $(q, g_\Pi(r))$-FPT gap reduction from Clique to Maximum Induced Subgraph with property Π. Hence, by applying Proposition 3.6, we immediately arrive at the following corollary.

Corollary 5.11. *Assuming Gap-ETH, for any property Π that contains all independent sets but not all cliques (or vice versa), Maximum Induced Subgraph with property Π is $\Omega(g_\Pi)$-weakly inherently enumerative where g_Π is the function from Theorem 5.9.*

We remark here that, for some properties, g_Π can be much larger than the bound given by the Ramsey's Theorem; for instance, if Π is planarity, then the Ramsey's Theorem only gives $g_\Pi(r) = \Omega(r^{1/5})$ but it is easy to see that, for planar graphs, there always exist an independent set of linear size and $g_\Pi(r)$ is hence as large as $\Omega(r)$.

5.4 Maximum Balanced Biclique, Maximum Induced Matching on Bipartite Graphs and Densest k-Subgraph

We next prove FPT inapproximability for the Maximum Balanced Biclique, Maximum Induced Matching on Bipartite Graphs and Densest k-Subgraph. Unlike the previous proofs, we will not reduce from any label cover problem. The starting point for the results in this section will instead be a recent construction of Manurangsi for the ETH-hardness of Densest k-Subgraph [Man17a]. By interpreting this construction in a different perspective, we can modify it in such a way that we arrive at a stronger form of inherently enumerative hardness for Clique. More specifically, the main theorem of this section is the following theorem, which is a stronger form of Theorem 5.1 in that the soundness not only rules out cliques, but also rules out bicliques as well.

Theorem 5.12. *Assuming Gap-ETH, there exist constants $\delta, \rho > 0$ such that, for any positive integers $q \geq r \geq \rho$, no algorithm can take a graph G and distinguish between the following cases in $O_{q,r}(|V(G)|^{\delta\sqrt{r}})$ time:*

- $\mathsf{Clique}(G) \geq q$.
- $\mathsf{Biclique}(G) < r$.

The weakly inherently enumerativeness (and therefore totally FPT inapproximability) of Maximum Balanced Biclique and Maximum Induced Matching on Bipartite Graphs follows easily from Theorem 5.12. We will show these results in the subsequent subsections; for now, let us turn our attention to the proof of the theorem.

The main theorem of this section can be stated as follows.

Theorem 5.13. *For any $d, \varepsilon > 0$, there is a constant $\gamma = \gamma(d, \varepsilon) > 0$ such that there exists a (randomized) reduction that takes in a parameter r and a 3-SAT instance ϕ with n variables and m clauses where each variable appears in at most d constraints and produces a graph $G_{\phi,r} = (V_{\phi,r}, E_{\phi,r})$ such that, for any sufficiently large r (depending only on d, ε but not n), the following properties hold with high probability:*

- *(Size)* $N := |V_{\phi,r}| \leq 2^{O_{d,\varepsilon}(n/\sqrt{r})}$.
- *(Completeness) if* $\mathsf{SAT}(\phi) = m$, *then* $\mathsf{Clique}(G_{\phi,r}) \geq N^{\gamma/\sqrt{r}}$.
- *(Soundness) if* $\mathsf{SAT}(\phi) \leq (1-\varepsilon)m$, *then* $\mathsf{Biclique}(G_{\phi,r}) < r$.

It is not hard to see that, in the Gap-ETH assumption, we can, without loss of generality, assume that each variable appears in only a bounded number of clauses (See [MR16, p. 21]). Hence, Theorem 5.13 together with Gap-ETH implies Theorem 5.12.

As mentioned earlier, our result builds upon an intermediate lemma used in proving the hardness of approximating Densest k-Subgraph in [Man17a]. Due to this, it will be easier to describe our reduction in terms of the reduction from [Man17a]; in this regard, our reduction can be viewed as vertex subsampling (with appropriate probability) of the graph produced by the reduction from [Man17a]. The reduction is described formally in Figure 1. Note that the two parameters ℓ and p will be chosen as $\Theta_{d,\varepsilon}(n/\sqrt{r})$ and $2^{\Theta_{d,\varepsilon}(\ell^2/n)}/\binom{n}{\ell}$, respectively, where the constants in $\Theta_{d,\varepsilon}(\cdot)$ will be selected based on the parameters from the intermediate lemma in [Man17a].

The main lemma of [Man17a] is stated below. Roughly speaking, when $\mathsf{SAT}(\phi) \leq (1-\varepsilon)m$, the lemma gives an upper bound on the number of occurrences of $K_{t,t}$ for every $t > 0$. When p and t

Input: a 3-SAT instance ϕ and parameters $p \in (0,1)$ and $\ell \in \mathbb{N}$ such that $\ell \leq n$.
Output: a graph $G_{\phi,\ell,p} = (V_{\phi,\ell,p}, E_{\phi,\ell,p})$.
The graph $G_{\phi,\ell,p}$ is generated as follows.

- First, we create a graph $\widetilde{G}_{\phi,\ell} = (\widetilde{V}_{\phi,\ell}, \widetilde{E}_{\phi,\ell})$ as constructed in [Man17a]. More specifically, the vertex set $\widetilde{V}_{\phi,\ell}$ and the edge set $\widetilde{E}_{\phi,\ell}$ are defined as follows.
  - The vertex set $\widetilde{V}_{\phi,\ell}$ consists of all partial assignments of ℓ variables, i.e., $\widetilde{V}_{\phi,\ell} := \{\sigma : S \to \{0,1\} \mid S \in \binom{\mathcal{X}}{\ell}\}$ where $\mathcal{X}$ is the set of all variables in ϕ.
  - There exists an edge between two vertices $\sigma_1 : S_1 \to \{0,1\}$ and $\sigma_2 : S_2 \to \{0,1\} \in \widetilde{V}_{\phi,\ell}$ if and only if (1) they are consistent (i.e., $\sigma_1(S_1 \cap S_2) = \sigma_2(S_1 \cap S_2)$) and (2) the partial assignment induced by σ_1, σ_2 does not violate any constraint (i.e., every constraint that lies entirely inside $S_1 \cup S_2$ is satisfied by the partial assignment induced by σ_1, σ_2).
- Our graph $G_{\phi,\ell,p} = (V_{\phi,\ell,p}, E_{\phi,\ell,p})$ can then be easily generated as follows.
  - Let $V_{\phi,\ell,p}$ be a random subset of $\widetilde{V}_{\phi,\ell}$ such that each vertex $v \in \widetilde{V}_{\phi,\ell}$ is included independently and randomly in $V_{\phi,\ell,p}$ with probability p.
  - We connect $u, v \in V_{\phi,\ell,p}$ if and only if $(u,v) \in \widetilde{E}_{\phi,\ell}$.

Figure 1: The Reduction from Gap-3SAT to Maximum Balanced Biclique

are chosen appropriately, this implies that w.h.p. there is no t-biclique in our subsampled graph. Note that the size and completeness properties are obvious from the construction while the exact statement of the soundness can be found in the proof of Theorem 8 in [Man17a].

Lemma 5.14 ([Man17a])**.** *Let $d, \varepsilon, \phi, n, m, \ell$ be as in Theorem 5.13 and Figure 1. There is a constant $\delta, \lambda > 0$ depending only on d, ε such that, for any sufficiently large n, the graph $G_{\phi,\ell} = (V_{\phi,\ell}, E_{\phi,\ell})$ described in Figure 1 has the following properties*

- *(Size) $|V_{\phi,\ell}| = \binom{n}{\ell} 2^\ell$.*
- *(Completeness) if $\mathsf{SAT}(\phi) = m$, $\widetilde{G}_{\phi,\ell}$ contains a $\binom{n}{\ell}$-clique.*
- *(Soundness) if $\mathsf{SAT}(\phi) \leq (1-\varepsilon)m$, then $\widetilde{G}_{\phi,\ell}$ contains at most $2^{4n}(2^{-\lambda\ell^2/n}\binom{n}{\ell})^{2t}$ occurrences[9] of $K_{t,t}$ for any $t > 0$.*

Theorem 5.13 follows rather easily from the above lemma by choosing appropriate ℓ and p.

Proof of Theorem 5.13. We let $G_{\phi,r} = G_{\phi,\ell,p}$ from the reduction in Figure 1 with parameters $\ell = \frac{4n}{\sqrt{\lambda r}}$ and $p = 2^{\frac{\lambda\ell^2}{2n}} / \binom{n}{\ell}$. For convenience, we assume without loss of generality that $\lambda < 1$.

[9] We say that $S, T \subseteq V_{\phi,\ell}$ is an occurrence of $K_{t,t}$ if $|S| = |T| = t$, $S \cap T = \emptyset$ and, for every $s \in S, t \in T$, there is an edge between s and t in $G_{\phi,\ell}$. The number of occurrences of $K_{t,t}$ of $G_{\phi,\ell}$ is simply the number of such pairs (S,T)'s.

Size. Since each vertex in $V_{\phi,\ell}$ is included that $V_{\phi,\ell,p}$ independently with probability p, we have $\mathbb{E}[|V_{\phi,\ell,p}|] = p|V_{\phi,\ell}| = 2^{\ell+\frac{\lambda\ell^2}{2n}} \leq 2^{2\ell}$. Hence, from Chernoff bound, $|V_{\phi,\ell,p}| \leq 2^{10\ell} = 2^{\Omega_{d,\varepsilon}(n/\sqrt{r})}$ w.h.p.

Completeness. Suppose that ϕ is satisfiable. Let C be the clique of size $\binom{n}{\ell}$ in $\widetilde{G}_{\phi,\ell}$, which is guaranteed to exist by Lemma 5.14. From how $G_{\phi,\ell,p}$ is defined, $C \cap V_{\phi,\ell,p}$ induces a clique in $G_{\phi,\ell,p}$. Moreover, $\mathbb{E}[|C \cap V_{\phi,\ell,p}|] = p|C| = 2^{\frac{\lambda\ell^2}{2n}}$. Again, from Chernoff bound, $\mathsf{Clique}(G_{\phi,\ell,p}) \geq 2^{\frac{\lambda\ell^2}{2n}}$ w.h.p. Combined with the above bound on N, $\mathsf{Clique}(G_{\phi,\ell,p}) \geq N^{\gamma/\sqrt{r}}$ w.h.p. when $\gamma := \sqrt{\lambda}/20 = O_{d,\varepsilon}(1)$.

Soundness. Suppose that $\mathsf{SAT}(\phi) \leq (1-\varepsilon)m$. Consider any subsets $S, T \subseteq \widetilde{V}_{\phi,\ell}$ that is an occurrence of $K_{r,r}$ in $\widetilde{G}_{\phi,\ell}$. From how $G_{\phi,\ell,p}$ is defined, $\mathsf{Biclique}(G_{\phi,\ell,p}) \geq r$ if and only if, for at least one such pair (S,T), $S \cup T \subseteq V_{\phi,\ell,p}$. The probability of this event is bounded above by

$$\begin{aligned}\sum_{\substack{S,T \subseteq \widetilde{V}_{\phi,\ell} \\ S,T \text{ is an occurrence of } K_{r,r} \text{ in } \widetilde{G}_{\phi,\ell}}} \Pr[S,T \subseteq V_{\phi,\ell,p}] &\leq 2^{4n}\left(2^{-\lambda\ell^2/n}\binom{n}{\ell}\right)^{2r} \cdot p^{2r} \\ &= 2^{4n}\left(2^{-\frac{\lambda\ell^2}{2n}}\right)^{2r} \\ &= o(1).\end{aligned}$$

where the first inequality comes from the bound in the soundness of Lemma 5.14 and the fact that the sampling of each vertex is done independently.

As a result, the subsampled graph $G_{\phi,\ell,p}$ is $K_{r,r}$-free with high probability as desired. □

5.4.1 Maximum Balanced Biclique

We now give a simple reduction from the "Clique vs Biclique" problem (from Theorem 5.12) to the Maximum Balanced Biclique problem, which yields FPT inapproximability of the latter.

Lemma 5.15. *For any graph $G = (V,E)$, let $B_e[G] = (V_{B_e[G]}, E_{B_e[G]})$ be the bipartite graph whose vertex set is $V_{B_e[G]} := V \times [2]$ and two vertices $(u,i),(v,j)$ are connected by an edge if and only if $(u,v) \in E$ or $u = v$, and $i \neq j$. Then the following properties hold for any graph G.*

- $\mathsf{Biclique}(B_e[G]) \geq \mathsf{Clique}(G)$.
- $\mathsf{Biclique}(B_e[G]) \leq 2\mathsf{Biclique}(G) + 1$.

Proof. It is easy to see that $\mathsf{Biclique}(B_e[G]) \geq \mathsf{Clique}(G)$ since, for any $C \subseteq V$ that induces a clique in G, $C \times [2] \subseteq V_{B_e[G]}$ induces a $|C|$-biclique in $B_e[G]$.

To see that $\mathsf{Biclique}(B_e[G]) \leq 2\mathsf{Biclique}(G)+1$, consider any $S \subseteq V_{B_e[G]}$ that induces a k-biclique in $B_e[G]$. Note that S can be partitioned into $S_1 = S \cap (V \times \{1\})$ and $S_2 = S \cap (V \times \{2\})$.

Now consider the projections of S_1 and S_2 into $V(G)$, i.e., $T_1 = \{v : (v,1) \in S\}$ and $T_2 = \{v : (v,2) \in S\}$. Note that $|T_1| = |T_2| = k$. Since $S_1 \cup S_2$ induces a biclique in $B_e[G]$, we have, for every $u \in T_1$ and $v \in T_2$, either $u = v$ or $(u,v) \in E$. Observe that if there were no former case (i.e., $T_1 \cap T_2 = \emptyset$), then we would have a k-biclique in G. Even if $T_1 \cap T_2 \neq \emptyset$, we can still get back a $\lfloor k/2 \rfloor$-biclique in G by uncrossing the sets T_1 and T_2 in a natural way by assigning half of the

intersection to T_1 and the other half to T_2. To be formal, we partition $T_1 \cap T_2$ into roughly equal sets U_1 and U_2 (i.e., $||U_1| - |U_2|| \leq 1$), and we then define new sets T_1' and T_2' by

$$T_1' = (T_1 \setminus T_2) \cup U_1 \text{ and } T_2' = (T_2 \setminus T_1) \cup U_2.$$

It is not hard to see that G has an edge between every pair of vertices between T_1', T_2' and that $|T_1'|, |T_2'| \geq \lfloor k/2 \rfloor$. Thus, $\mathsf{Biclique}(G) \geq \lfloor k/2 \rfloor \geq (k-1)/2$. Therefore, $\mathsf{Biclique}(B_e[G]) \leq 2\mathsf{Biclique}(G) + 1$ as desired. □

Thanks to the above lemma, we can conclude that the reduction $G \mapsto B_e[G]$ is a $(2q, (r+1)/2)$-FPT gap reduction from the "Clique vs Biclique" problem to Maximum Balanced Biclique, although the former is not a well-defined optimization problem. Nevertheless, it is easy to check that a proof along the line of Proposition 3.6 still works and it gives the following result:

Corollary 5.16. *Assuming Gap-ETH, Maximum Balanced Biclique are $\Omega(\sqrt{r})$-weakly inherently enumerative and thus FPT inapproximable.*

It is worth noting that the Maximum Edge Biclique problem, a well-studied variant of the Maximum Balanced Biclique problem where the goal is to find a (not necessarily balanced) complete bipartite subgraph of a given bipartite graph that contains as many edges as possible, is in FPT. This is because the optimum is at least the maximum degree, but when the degree is bounded above by r, all bicliques can be enumerated in $2^{O(r)}\,\mathrm{poly}(n)$ time.

5.4.2 Maximum Induced Matching on Bipartite Graphs

Next, we prove the FPT hardness of approximation for the Maximum Induced Matching problem on bipartite graphs. Again, the proof will be a simple reduction from Theorem 5.12. The argument below is similar to that used in Lemma IV.4 of [CLN13b]. We include it here for completeness.

Lemma 5.17. *For any graph $G = (V, E)$, let $B_e[\bar{G}] = (V_{B_e[\bar{G}]}, E_{B_e[\bar{G}]})$ be the bipartite graph whose vertex is $V_{B_e[\bar{G}]} := V \times [2]$ and two vertices $(u, i), (v, j)$ are connected by an edge if and only if $(u, v) \notin E$ or $u = v$, and $i \neq j$. Then, the following properties hold for any graph G.*

- $\mathsf{IM}(B_e[\bar{G}]) \geq \mathsf{Clique}(G)$.
- $\mathsf{IM}(B_e[\bar{G}]) \leq 2\mathsf{Biclique}(G) + 1$.

Proof. Consider any $S \subseteq V$ that induces a clique in G. It is obvious that $S \times [2] \subseteq V_{B_e[\bar{G}]}$ induces a matching in $B_e[\bar{G}]$.

Next, consider any induced matching $\{(u_1, v_1), \ldots, (u_m, v_m)\}$ of size m. Assume w.l.o.g. that $u_1, \ldots, u_m \in V \times \{1\}$ and $v_1, \ldots, v_m \in V \times \{2\}$. Define $\pi_1 : V \times [2] \to V$ to be a projection operator that projects on to the first coordinate.

Let $S_1 = \pi_1(\{u_1, \ldots, u_{\lfloor m/2 \rfloor}\})$ and $S_2 = \pi_1(\{v_{\lceil m/2 \rceil + 1}, \ldots, v_m\})$. From the definition of $B_e[\bar{G}]$ and from the fact that there is no edge between $(S_1 \times \{1\})$ and $(S_2 \times \{2\})$, it is easy to check that $S_1 \cap S_2 = \emptyset$ and, for every $u \in S_1$ and $v \in S_2$, $(u, v) \in E$. In other words, (S_1, S_2) is an occurrence of $\lfloor m/2 \rfloor$ in G. Hence, we conclude that $\mathsf{IM}(B_e[\bar{G}]) \leq 2\mathsf{Biclique}(G) + 1$. □

Similar to Biclique, it is easy to see that the above reduction implies the following running time lower bound and FPT inapproximability for Maximum Induced Matching on Bipartite Graphs.

Corollary 5.18. *Assuming Gap-ETH, Maximum Induced Matching on Bipartite Graphs are $\Omega(\sqrt{r})$-weakly inherently enumerative and thus FPT inapproximable.*

5.4.3 Densest k-Subgraph

Finally, we will show FPT inapproximability result for Densest k-Subgraph. Alas, we are not able to show $o(k)$-ratio FPT inapproximability, which would have been optimal since the trivial algorithm gives an $O(k)$-approximation for the problem. Nonetheless, we will show an $k^{o(1)}$-factor FPT inapproximability for the problem. We note that below we will state the result as if k is the parameter. This is the same as using the optimum as the parameter since (in the non-trivial case) the optimum is always between $\lfloor k/2 \rfloor$ and $\binom{k}{2}$ (inclusive).

To derive our result, we resort to a well-known result in extremal combinatorics called the Kővári-Sós-Turán (KST) Theorem, which basically states that if a graph contains no small bicliques, then it is sparse. The KST theorem is stated formally below.

Theorem 5.19 (Kővári-Sós-Turán (KST) Theorem [KST54])**.** *For every positive integer n and $t \leq n$, every $K_{t,t}$-free graph on n vertices has at most $O(n^{2-1/t})$ edges (i.e., density $O(n^{-1/t})$).*

We remark that a generalization of the KST Theorem was also a crucial ingredient in the proof of ETH-hardness of approximating Densest k-Subgraph in [Man17a]. The situation is simpler for us here since we can simply apply the KST Theorem to Theorem 5.12, which yields the following theorem.

Theorem 5.20. *Assuming Gap-ETH, there exist a constant $\delta > 0$ and an integer $\rho > 0$ such that, for any integer $q \geq r \geq \rho$, no algorithm can take a graph $G = (V, E)$ and distinguish between the following cases in $O_{q,r}(|V|^{\delta\sqrt{r}})$ time:*

- $\mathsf{Den}_q(G) = 1$.
- $\mathsf{Den}_q(G) < O(q^{-1/r})$.

From the above theorem, it is easy to show the $k^{o(1)}$-factor FPT inapproximability of Densest k-Subgraph as formalized below. We note here that our result applies to a special case of Densest k-Subgraph in which the input graph is promised to contain a k-clique; this problem is sometimes referred to as *Densest k-Subgraph with perfect completeness* [BKRW17; Man17a].

Lemma 5.21. *Assuming Gap-ETH, for every function $f = o(1)$ and every function t, there is no $t(k) \cdot n^{O(1)}$-time algorithm such that, given an integer k and any graph $G = (V, E)$ on n vertices that contains at least one k-clique, always outputs $S \subseteq V$ of size k such that $\mathsf{Den}(S) \geq k^{-f(k)}$.*

Proof. Suppose for the sake of contradiction that there is a $t(k) \cdot |V|^D$-time algorithm $\mathbb{A}$ that, given an integer k and any graph $G = (V, E)$ that contains a k-clique, always outputs $S \subseteq V$ of size k such that $\mathsf{Den}(S) \geq k^{-f(k)}$ for some function $f = o(1)$, some function t and some constant $D > 0$.

Let $r = \max\{\lceil \rho \rceil, \lceil (D/\delta)^2 \rceil\}$ where ρ is the constant from Theorem 5.20. Note that $O(q^{-1/r}) = q^{O(1)/\log q - 1/r}$. Now, since $\lim_{q\to\infty} f(q) + O(1)/\log q = 0$, there exists a sufficiently large q such that the term $O(q^{-1/r})$ is less than $q^{-f(q)}$. In other words, $\mathbb{A}$ can distinguish between the two cases in Theorem 5.20 in time $t(q) \cdot n^D = O_{q,r}(|V|^{\delta\sqrt{r}})$, which would break Gap-ETH. □

6 Conclusion and Discussions

In this paper, we prove that Clique and DomSet are totally FPT inapproximable. In fact, we show a stronger property that they are inherently enumerative, i.e., the best way to approximate

both problems is to essentially enumerate all possibilities. Since Clique and DomSet are complete problems for the class W[1] and W[2], respectively, it might be possible that these two problems can be sources of FPT-inapproximabilities for many other problems that admit no FPT algorithms.

We would like to also mention that there are some problems that are known to be totally FPT-inapproximable under weaker assumptions. Examples of such problems are *independent dominating set* and *induced path*. The former has been shown to be FPT-inapproximable under the assumption FPT $\neq$ W[2] in [DFMR08]. For the induced path problem, we show in Appendix C that it is FPT-inapproximable under the assumption FPT $\neq$ W[1]. It would be interesting to understand whether it is possible to also base the total FPT-inapproximabilities of Clique and DomSet under assumptions that are weaker than Gap-ETH, such as FPT $\neq$ W[1], FPT $\neq$ W[2] or ETH. As discussed in the introduction, it was recently shown in [KLM17] that DomSet is totally FPT inapproximable under FPT $\neq$ W[1], and the more refined running-time lower bounds were also shown under ETH and SETH. Nevertheless, we are not aware of any FPT inapproximability result for Clique under an assumption weaker than Gap-ETH.

Another interesting further research direction is to study the trade-off between the running time and the approximation ratio of problems that are known to be FPT-approximable or admit FPT (exact) algorithms. The exploration of such trade-off may be useful in both theory and practice.

Acknowledgment

We thank Benny Applebaum for sharing with us his result [App17]. Pasin would like to thank Igor Shinkar and Daniel Reichman for discussions on a related problem that inspires part of the proof, and Aviad Rubinstein for useful discussions regarding FPT inapproximability. We also thank Igor for pointing us to [Kay14]. Danupon and Parinya would like to thank Per Austrin for discussions. Parinya would also like to thank Nikhil Bansal, Jesper Nederlof, Karl Bringmann and Holger Dell for insightful discussions. Bundit would like to thank Uriel Feige for useful discussions on Clique.

Marek Cygan is supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 677651). Guy Kortsarz is supported in part by NSF grants 1218620 and 1540547. Bundit Laekhanukit is partially supported by ISF Grant No. 621/12 and I-CORE Grant No. 4/11. Danupon Nanongkai is supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 715672 and the Swedish Research Council (Reg. No. 2015-04659). Pasin Manurangsi and Luca Trevisan are supported by NSF Grants No. CCF 1540685 and CCF 1655215.

The preliminary version of this paper was done while Bundit Laekhanukit was at the Weizmann Institute of Science, and some parts of the works were done while Parinya Chalermsook and Bundit Laekhanukit were visiting the Simons Institute for the Theory of Computing under the support of the DIMACS/Simons Collaboration on Bridging Continuous and Discrete Optimization through NSF Grant No. CCF-1740425.

References

[AAMMW11] Noga Alon, Sanjeev Arora, Rajsekar Manokaran, Dana Moshkovitz, and Omri Weinstein. "Inapproximabilty of Densest k-Subgraph from Average Case Hardness". Unpublished Manuscript. 2011 (cit. on p. 6).

[AIM14] Scott Aaronson, Russell Impagliazzo, and Dana Moshkovitz. "AM with Multiple Merlins". In: *CCC*. 2014, pp. 44–55 (cit. on p. 40).

[ALMSS98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. "Proof Verification and the Hardness of Approximation Problems". In: *J. ACM* 45.3 (1998), pp. 501–555 (cit. on pp. 5, 41).

[AOW15] Sarah R. Allen, Ryan O'Donnell, and David Witmer. "How to Refute a Random CSP". In: *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*. 2015, pp. 689–708 (cit. on p. 42).

[ARW17] Amir Abboud, Aviad Rubinstein, and R. Ryan Williams. "Distributed PCP Theorems for Hardness of Approximation in P". In: *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*. 2017, pp. 25–36 (cit. on p. 5).

[AS98] Sanjeev Arora and Shmuel Safra. "Probabilistic Checking of Proofs: A New Characterization of NP". In: *J. ACM* 45.1 (1998), pp. 70–122 (cit. on pp. 5, 41).

[App17] Benny Applebaum. "Exponentially-Hard gap-CSP and local PRG via Local Hardcore Functions". In: *ECCC* 24 (2017), p. 63 (cit. on pp. 30, 42).

[BCCFV10] 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*. 2010, pp. 201–210 (cit. on p. 6).

[BCVGZ12] Aditya Bhaskara, Moses Charikar, Aravindan Vijayaraghavan, Venkatesan Guruswami, 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*. SODA '12. Kyoto, Japan: Society for Industrial and Applied Mathematics, 2012, pp. 388–405 (cit. on p. 6).

[BEKP15] Edouard Bonnet, Bruno Escoffier, Eun Jung Kim, and Vangelis Th. Paschos. "On Subexponential and FPT-Time Inapproximability". In: *Algorithmica* 71.3 (2015), pp. 541–565 (cit. on pp. i, 1, 3, 4).

[BGHKK16] Amey Bhangale, Rajiv Gandhi, Mohammad Taghi Hajiaghayi, Rohit Khandekar, and Guy Kortsarz. "Bicovering: Covering Edges With Two Small Subsets of Vertices". In: *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*. 2016, 6:1–6:12 (cit. on p. 6).

[BGLR93] Mihir Bellare, Shafi Goldwasser, Carsten Lund, and A. Russeli. "Efficient probabilistically checkable proofs and applications to approximations". In: *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*. 1993, pp. 294–304 (cit. on p. 5).

[BGS98] Mihir Bellare, Oded Goldreich, and Madhu Sudan. "Free Bits, PCPs, and Nonapproximability-Towards Tight Results". In: *SIAM J. Comput.* 27.3 (1998), pp. 804–915 (cit. on pp. 5, 15).

[BKRW17] Mark Braverman, Young Kun-Ko, Aviad Rubinstein, and Omri Weinstein. "ETH Hardness for Densest-k-Subgraph with Perfect Completeness". In: *SODA*. 2017, pp. 1326–1341 (cit. on pp. 6, 29).

[BLP16] Édouard Bonnet, Michael Lampis, and Vangelis Th. Paschos. "Time-Approximation Trade-offs for Inapproximable Problems". In: *STACS*. 2016, 22:1–22:14 (cit. on p. 2).

[BS92] Piotr Berman and Georg Schnitger. "On the Complexity of Approximating the Independent Set Problem". In: *Inf. Comput.* 96.1 (1992), pp. 77–94 (cit. on pp. 2, 3, 5, 15).

[BS94] Mihir Bellare and Madhu Sudan. "Improved non-approximability results". In: *STOC*. 1994, pp. 184–193 (cit. on p. 5).

[CCKLMNT17] Parinya Chalermsook, Marek Cygan, Guy Kortsarz, Bundit Laekhanukit, Pasin Manurangsi, Danupon Nanongkai, and Luca Trevisan. "From Gap-ETH to FPT-Inapproximability: Clique, Dominating Set, and More". In: *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*. 2017, pp. 743–754 (cit. on p. i).

[CGG06] Yijia Chen, Martin Grohe, and Magdalena Grüber. "On Parameterized Approximability". In: *IWPEC*. 2006, pp. 109–120 (cit. on pp. 7, 37).

[CHK11] Moses Charikar, MohammadTaghi Hajiaghayi, and Howard J. Karloff. "Improved Approximation Algorithms for Label Cover Problems". In: *Algorithmica* 61.1 (2011), pp. 190–206 (cit. on p. 13).

[CHK13] Rajesh Hemant Chitnis, MohammadTaghi Hajiaghayi, and Guy Kortsarz. "Fixed-Parameter and Approximation Algorithms: A New Look". In: *IPEC*. 2013, pp. 110–122 (cit. on p. 4).

[CHKX04] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. "Linear FPT reductions and computational lower bounds". In: *STOC*. 2004, pp. 212–221 (cit. on pp. 3, 4).

[CHKX06a] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. "On the computational hardness based on linear FPT-reductions". In: *J. Comb. Optim.* 11.2 (2006), pp. 231–247 (cit. on pp. 2–4).

[CHKX06b] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. "Strong computational lower bounds via parameterized complexity". In: *J. Comput. Syst. Sci.* 72.8 (2006), pp. 1346–1367 (cit. on p. 2).

[CL16] Yijia Chen and Bingkai Lin. "The Constant Inapproximability of the Parameterized Dominating Set Problem". In: *FOCS*. 2016, pp. 505–514 (cit. on pp. i, 2–4).

[CLN13a] Parinya Chalermsook, Bundit Laekhanukit, and Danupon Nanongkai. "Graph Products Revisited: Tight Approximation Hardness of Induced Matching, Poset Dimension and More". In: *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*. 2013, pp. 1557–1576 (cit. on p. 6).

[CLN13b] Parinya Chalermsook, Bundit Laekhanukit, and Danupon Nanongkai. "Independent Set, Induced Matching, and Pricing: Connections and Tight (Subexponential Time) Approximation Hardnesses". In: *FOCS*. 2013, pp. 370–379 (cit. on p. 28).

[CMMV17] Eden Chlamtác, Pasin Manurangsi, Dana Moshkovitz, and Aravindan Vijayaraghavan. "Approximation Algorithms for Label Cover and The Log-Density Threshold". In: *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*. 2017, pp. 900–919 (cit. on p. 6).

[Cam89] Kathie Cameron. "Induced matchings". In: *Discrete Applied Mathematics* 24.1-3 (1989), pp. 97–102 (cit. on p. 6).

[Chv79] Vasek Chvátal. "A Greedy Heuristic for the Set-Covering Problem". In: *Math. Oper. Res.* 4.3 (1979), pp. 233–235 (cit. on p. 5).

[DF13] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013 (cit. on pp. i, 1).

[DFMR08] Rodney G. Downey, Michael R. Fellows, Catherine McCartin, and Frances A. Rosamond. "Parameterized approximation of dominating set problems". In: *Inf. Process. Lett.* 109.1 (2008), pp. 68–70 (cit. on p. 30).

[DMZ05] William Duckworth, David Manlove, and Michele Zito. "On the approximability of the maximum induced matching problem". In: *J. Discrete Algorithms* 3.1 (2005), pp. 79–91 (cit. on p. 6).

[DS14] Irit Dinur and David Steurer. "Analytical approach to parallel repetition". In: *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*. 2014, pp. 624–633 (cit. on p. 5).

[Din07] Irit Dinur. "The PCP theorem by gap amplification". In: *J. ACM* 54.3 (2007), p. 12 (cit. on p. 41).

[Din16] Irit Dinur. "Mildly exponential reduction from gap 3SAT to polynomial-gap label-cover". In: *ECCC* 23 (2016), p. 128 (cit. on pp. i, 2, 9, 41, 42).

[EH00] Lars Engebretsen and Jonas Holmerin. "Clique Is Hard to Approximate within $n^{1\text{-}o(1)}$". In: *Automata, Languages and Programming, 27th International Colloquium, ICALP 2000, Geneva, Switzerland, July 9-15, 2000, Proceedings*. 2000, pp. 2–12 (cit. on p. 5).

[ERRS09] Khaled M. Elbassioni, Rajiv Raman, Saurabh Ray, and René Sitters. "On the approximability of the maximum feasible subsystem problem with 0/1-coefficients". In: *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*. 2009, pp. 1210–1219 (cit. on p. 6).

[FGLSS96] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. "Interactive Proofs and the Hardness of Approximating Cliques". In: *J. ACM* 43.2 (1996), pp. 268–292 (cit. on pp. 3, 5, 20).

[FGMS12] Michael R. Fellows, Jiong Guo, Dániel Marx, and Saket Saurabh. "Data Reduction and Problem Kernels (Dagstuhl Seminar 12241)". In: *Dagstuhl Reports* 2.6 (2012). Ed. by Michael R. Fellows, Jiong Guo, Dániel Marx, and Saket Saurabh, pp. 26–50. ISSN: 2192-5283 (cit. on pp. i, 1).

[FK00] Uriel Feige and Joe Kilian. "Two-Prover Protocols - Low Error at Affordable Rates". In: *SIAM J. Comput.* 30.1 (2000), pp. 324–346 (cit. on p. 5).

[FK05] Uriel Feige and Shimon Kogan. "The hardness of approximating hereditary properties". Technical Report. 2005 (cit. on pp. 5, 6).

[FKP01] Uriel Feige, Guy Kortsarz, and David Peleg. "The Dense k-Subgraph Problem". In: *Algorithmica* 29.3 (2001), pp. 410–421 (cit. on p. 6).

[Fei02] Uriel Feige. "Relations between average case complexity and approximation complexity". In: *STOC*. 2002, pp. 534–543 (cit. on p. 6).

[Fei04] Uriel Feige. "Approximating Maximum Clique by Removing Subgraphs". In: *SIAM J. Discrete Math.* 18.2 (2004), pp. 219–225 (cit. on pp. 1, 5).

[Fei98] Uriel Feige. "A Threshold of ln n for Approximating Set Cover". In: *J. ACM* 45.4 (1998), pp. 634–652 (cit. on pp. 3–5, 22).

[GG07] Martin Grohe and Magdalena Grüber. "Parameterized Approximability of the Disjoint Cycle Problem". In: *ICALP*. 2007 (cit. on p. 1).

[Gri01] Dima Grigoriev. "Linear lower bound on degrees of Positivstellensatz calculus proofs for the parity". In: *Theor. Comput. Sci.* 259.1-2 (2001), pp. 613–622 (cit. on p. 42).

[HKK13] Mohammad Taghi Hajiaghayi, Rohit Khandekar, and Guy Kortsarz. "Fixed Parameter Inapproximability for Clique and Set Cover in Time Super-exponential in OPT". In: *CoRR* abs/1310.2711 (2013) (cit. on pp. i, 1–4).

[Hal00] Magnús M. Halldórsson. "Approximations of Weighted Independent Set and Hereditary Subset Problems". In: *J. Graph Algorithms Appl.* 4.1 (2000) (cit. on p. 6).

[Hås01] Johan Håstad. "Some optimal inapproximability results". In: *J. ACM* 48.4 (2001), pp. 798–859 (cit. on p. 41).

[Hås96] Johan Håstad. "Clique is Hard to Approximate Within $n^{1-\epsilon}$". In: *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*. 1996, pp. 627–636 (cit. on p. 5).

[IP01] Russell Impagliazzo and Ramamohan Paturi. "On the Complexity of k-SAT". In: *J. Comput. Syst. Sci.* 62.2 (2001), pp. 367–375 (cit. on p. 9).

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. "Which Problems Have Strongly Exponential Complexity?" In: *J. Comput. Syst. Sci.* 63.4 (2001), pp. 512–530 (cit. on p. 9).

[KLM17] Karthik C. S., Bundit Laekhanukit, and Pasin Manurangsi. "On the Parameterized Complexity of Approximating Dominating Set". In: *CoRR* abs/1711.11029 (2017) (cit. on pp. 4, 5, 30).

[KP06] Subhash Khot and Ashok Kumar Ponnuswami. “Better Inapproximability Results for MaxClique, Chromatic Number and Min-3Lin-Deletion”. In: *ICALP*. 2006, pp. 226–237 (cit. on p. 5).

[KP93] Guy Kortsarz and David Peleg. “On Choosing a Dense Subgraph (Extended Abstract)”. In: *34th Annual Symposium on Foundations of Computer Science, Palo Alto, California, USA, 3-5 November 1993*. 1993, pp. 692–701 (cit. on p. 6).

[KR00] Subhash Khot and Venkatesh Raman. “Parameterized Complexity of Finding Subgraphs with Hereditary Properties”. In: *COCOON*. 2000, pp. 137–147 (cit. on pp. i, 3, 24).

[KS16] Subhash Khot and Igor Shinkar. “On Hardness of Approximating the Parameterized Clique Problem”. In: *ITCS*. 2016, pp. 37–45 (cit. on p. 1).

[KST54] Tamás Kővári, Vera T. Sós, and Pál Turán. “On a problem of K. Zarankiewicz”. eng. In: *Colloquium Mathematicae* 3.1 (1954), pp. 50–57 (cit. on p. 29).

[Kay14] Neeraj Kayal. *Solvability of Systems of Polynomial Equations over Finite Fields*. A talk given by Neeraj Kayal at the Simons Institute for the Theory of Computing, Berkeley, CA [Accessed: 2017/20/7]. Oct. 2014 (cit. on pp. 2, 30).

[Kho01] Subhash Khot. “Improved Inaproximability Results for MaxClique, Chromatic Number and Approximate Graph Coloring”. In: *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*. 2001, pp. 600–609 (cit. on p. 5).

[Kho06] Subhash Khot. “Ruling Out PTAS for Graph Min-Bisection, Dense k-Subgraph, and Bipartite Clique”. In: *SIAM J. Comput.* 36.4 (2006), pp. 1025–1071 (cit. on p. 6).

[LRS15] James R. Lee, Prasad Raghavendra, and David Steurer. “Lower Bounds on the Size of Semidefinite Programming Relaxations”. In: *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*. 2015, pp. 567–576 (cit. on p. 42).

[LY80] John M. Lewis and Mihalis Yannakakis. “The Node-Deletion Problem for Hereditary Properties is NP-Complete”. In: *J. Comput. Syst. Sci.* 20.2 (1980), pp. 219–230 (cit. on p. 5).

[LY93] Carsten Lund and Mihalis Yannakakis. “The Approximation of Maximum Subgraph Problems”. In: *Automata, Languages and Programming, 20nd International Colloquium, ICALP93, Lund, Sweden, July 5-9, 1993, Proceedings*. 1993, pp. 40–51 (cit. on p. 5).

[LY94] Carsten Lund and Mihalis Yannakakis. “On the Hardness of Approximating Minimization Problems”. In: *J. ACM* 41.5 (1994), pp. 960–981 (cit. on pp. 4, 5).

[Lin15] Bingkai Lin. “The Parameterized Complexity of k-Biclique”. In: *SODA*. 2015, pp. 605–615 (cit. on pp. i, 3, 10).

[MR16] Pasin Manurangsi and Prasad Raghavendra. “A Birthday Repetition Theorem and Complexity of Approximating Dense CSPs”. In: *CoRR* abs/1607.02986 (2016) (cit. on pp. i, 2, 9, 25, 41, 42).

[MS09] Hannes Moser and Somnath Sikdar. "The parameterized complexity of the induced matching problem". In: *Discrete Applied Mathematics* 157.4 (2009), pp. 715–727 (cit. on pp. i, 3).

[Man15] Pasin Manurangsi. "On Approximating Projection Games". MA thesis. Massachusetts Institute of Technology, 2015 (cit. on p. 6).

[Man17a] Pasin Manurangsi. "Almost-polynomial ratio ETH-hardness of approximating densest k-subgraph". In: *STOC*. 2017, pp. 954–961 (cit. on pp. 2, 3, 6, 25, 26, 29).

[Man17b] Pasin Manurangsi. "Inapproximability of Maximum Edge Biclique, Maximum Balanced Biclique and Minimum k-Cut from the Small Set Expansion Hypothesis". In: *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*. 2017, 79:1–79:14 (cit. on p. 6).

[Mar08] Dániel Marx. "Parameterized Complexity and Approximation Algorithms". In: *Comput. J.* 51.1 (2008), pp. 60–78 (cit. on pp. i, 1).

[Mar13] Dániel Marx. "Completely inapproximable monotone and antimonotone parameterized problems". In: *J. Comput. Syst. Sci.* 79.1 (2013), pp. 144–151 (cit. on p. 1).

[Mos15] Dana Moshkovitz. "The Projection Games Conjecture and the NP-Hardness of ln n-Approximating Set-Cover". In: *Theory of Computing* 11 (2015), pp. 221–235 (cit. on pp. 4, 5).

[PW10] Mihai Patrascu and Ryan Williams. "On the Possibility of Faster SAT Algorithms". In: *SODA*. 2010, pp. 1065–1075 (cit. on pp. 2–4).

[Rag08] Prasad Raghavendra. "Optimal algorithms and inapproximability results for every CSP?" In: *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*. 2008, pp. 245–254 (cit. on p. 42).

[Raz98] Ran Raz. "A Parallel Repetition Theorem". In: *SIAM J. Comput.* 27.3 (1998), pp. 763–803 (cit. on p. 2).

[SV82] Larry J. Stockmeyer and Vijay V. Vazirani. "NP-Completeness of Some Generalizations of the Maximum Matching Problem". In: *Inf. Process. Lett.* 15.1 (1982), pp. 14–19 (cit. on p. 6).

[Sch08] Grant Schoenebeck. "Linear Level Lasserre Lower Bounds for Certain k-CSPs". In: *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*. 2008, pp. 593–602 (cit. on p. 42).

[Vad12] Salil P. Vadhan. "Pseudorandomness". In: *Foundations and Trends in Theoretical Computer Science* 7.1-3 (2012), pp. 1–336 (cit. on p. 15).

[WS11] David P Williamson and David B Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011 (cit. on p. 13).

[Zuc07] David Zuckerman. "Linear Degree Extractors and the Inapproximability of Max Clique and Chromatic Number". In: *Theory of Computing* 3.1 (2007), pp. 103–128 (cit. on pp. 5, 15).

[Zuc96a] David Zuckerman. "On Unapproximable Versions of NP-Complete Problems". In: *SIAM J. Comput.* 25.6 (1996), pp. 1293–1304 (cit. on pp. 3, 15).

[Zuc96b] David Zuckerman. "Simulating BPP Using a General Weak Random Source". In: *Algorithmica* 16.4/5 (1996), pp. 367–391 (cit. on pp. 3, 15).

A Gap Problems vs Approximation Algorithms

In this section, we establish the connections between gap problems and FPT approximation algorithm by proving Proposition 2.3 and Proposition 2.4. Proposition 2.3 is in fact implied by a result due to Chen et al. [CGG06, Proposition 4]. We provide the proof here for completeness. In contrast, we are not aware of any prior proof of Proposition 2.4.

Proof of Proposition 2.3. We prove by contrapositive. Suppose that (2) is false, i.e., there exist computable functions $t : \mathbb{N} \to \mathbb{N}, f : \mathbb{N} \to [1, \infty)$ and an algorithm $\mathbb{B}$ such that, for every instance I of Π, the algorithm $\mathbb{B}$ runs in time $t(\mathsf{OPT}_\Pi(I)) \cdot |I|^D$ on the input I for some constant $D > 0$ and outputs $y \in \mathsf{SOL}_\Pi(I)$ of cost at most $\mathsf{OPT}_\Pi(I) \cdot f(\mathsf{OPT}_\Pi(I))$.

Let $t' : \mathbb{N} \to \mathbb{N}$ and $f' : \mathbb{N} \to [1, \infty)$ be functions that are defined by $t'(k) = \max_{i=1,\ldots,k} t(i)$ and $f'(k) = \max_{i=1,\ldots,k} f(i)$. Since t and f are computable, t' and f' are also computable.

Let $\mathbb{A}$ be an algorithm that takes an instance I of Π and a positive integer k and then works as follows. The algorithm $\mathbb{A}$ simulates an execution of $\mathbb{B}$ on I step-by-step. If $\mathbb{B}(I)$ does not finish within $t'(k) \cdot |I|^D$ time steps, then $\mathbb{A}$ terminates the execution and returns 0. Otherwise, let y be the output of $\mathbb{B}(I)$. Then algorithm $\mathbb{A}$ computes $\mathsf{COST}_\Pi(I, y)$ and then returns 1 if this value is at most $k \cdot f'(k)$; otherwise, $\mathbb{A}$ returns 0.

We claim that $\mathbb{A}$ is an f'-FPT gap approximation algorithm of Π. To see that this is the case, first notice that the running time of $\mathbb{A}$ is $O(t'(k) \cdot |I|^D + |I|^{O(1)})$ where $|I|^{O(1)}$ denotes the time used to compute the solution cost. Moreover, if $\mathsf{OPT}_\Pi(I) > k \cdot f'(k)$, then it is obvious to see that $\mathbb{A}$ always outputs 0. Finally, if $\mathsf{OPT}_\Pi(I) \leq k$, then, by our assumption on $\mathbb{B}$ and the definitions of t' and f', $\mathbb{B}(I)$ finishes in time $t(\mathsf{OPT}_\Pi(I)) \cdot |I|^D \leq t'(k) \cdot |I|^D$ and the output solution y has cost at most $\mathsf{OPT}_\Pi(I) \cdot f(\mathsf{OPT}_\Pi(I)) \leq k \cdot f'(k)$. Hence, $\mathbb{A}$ always outputs 1 in this case.

As a result, $\mathbb{A}$ is an f'-FPT gap approximation algorithm for Π, which concludes our proof. □

Proof of Proposition 2.4. We again prove by contrapositive. Suppose that (2) is false, i.e., there exist computable functions $t : \mathbb{N} \to \mathbb{N}, f : \mathbb{N} \to [1, \infty)$ such that $k/f(k)$ is non-decreasing and $\lim_{k\to\infty} k/f(k) = \infty$, and an algorithm $\mathbb{B}$ such that, for every instance I of Π, $\mathbb{B}$ runs in time $t(\mathsf{OPT}_\Pi(I)) \cdot |I|^D$ on the input I for some constant $D > 0$ and outputs $y \in \mathsf{SOL}_\Pi(I)$ of cost at least $\mathsf{OPT}_\Pi(I)/f(\mathsf{OPT}_\Pi(I))$.

Let $t' : \mathbb{N} \to \mathbb{N}$ be a function defined by $t'(k) = \max_{i=1,\ldots,k} t(i)$. Then clearly, t' is computable.

Let $\mathbb{A}$ be an algorithm that takes an instance I of Π and a positive integer k and then works as follows. The algorithm $\mathbb{A}$ simulates an execution of $\mathbb{B}$ on I step-by-step. If $\mathbb{B}(I)$ does not finish within $t'(k) \cdot |I|^D$ time steps, then $\mathbb{A}$ terminates the execution and returns 1. Otherwise, let y be the output of $\mathbb{B}(I)$. $\mathbb{A}$ computes $\mathsf{COST}_\Pi(I, y)$. The algorithm $\mathbb{A}$ then returns 1 if this value is at least $k/f(k)$; otherwise, $\mathbb{A}$ returns 0.

We claim that $\mathbb{A}$ is an f-FPT gap approximation algorithm of Π. To see that this is the case, first notice that the running time of $\mathbb{A}$ is $O(t'(k)\cdot|I|^D+|I|^{O(1)})$ where $|I|^{O(1)}$ denotes the time used to compute the solution cost. Moreover, if $\mathsf{OPT}_\Pi(I) < k/f'(k)$, then the running time of $\mathbb{B}(I)$ is at most $t(\mathsf{OPT}_\Pi(I))\cdot|I|^D \le t'(k)\cdot|I|^D$, which implies that $\mathbb{A}$ returns 0.

Suppose, on the other hand, that $\mathsf{OPT}_\Pi(I) \ge k$. If $\mathbb{B}(I)$ finishes in time $t'(k)\cdot|I|^D$, then, from the guarantee of $\mathbb{B}$, it must output $y \in \mathsf{SOL}_\Pi(I)$ with $\mathsf{COST}_\Pi(I,y) \ge \mathsf{OPT}_\Pi(I)/f(\mathsf{OPT}_\Pi(I))$, which is at least $k/f(k)$ since $k/f(k)$ is non-decreasing. Furthermore, if $\mathbb{B}(I)$ does not finish in the specified time, then $\mathbb{A}$ also returns 1 as desired.

As a result, $\mathbb{A}$ is an f-FPT gap approximation algorithm for Π, which concludes our proof. □

B Totally FPT Inapproximable Through FPT Gap Reductions (Proof of Proposition 3.5)

We will only show the proof when both Π_0 and Π_1 are maximization problems. Other cases can be proved analogously and therefore omitted.

We assume that (i) holds and will show that if the "then" part does not hold, then (ii) also does not hold. Recall from Definition 3.4 that (i) implies that there exist $C, D > 0$ such that the reduction from Π_0 (with parameters q and r) to Π_1 takes $O_{q,r}(|I_0|^C)$ time and always outputs an instance I_1 of size at most $O_{q,r}(|I_0|^D)$ on every input instance I_0. Now assume that the "then" part does *not* hold, i.e., Π_1 admits a $(t(k)|I_1|^F)$-time h-FPT gap approximation algorithm $\mathbb{A}$ for some function $h(k) = o(k)$ and constant F. We will show the following claim which says that (ii) does not hold (by Definition 2.1).

Claim B.1. *There exists a function $k \ge g'(k) = \omega(1)$ and an algorithm $\mathbb{B}$ that takes any input instance I_0 of the problem Π_0 and an integer k, and in $O_k(|I_0|^{O(1)})$ time can distinguish between $\mathsf{OPT}_{\Pi_0}(I_0) \ge k$ and $\mathsf{OPT}_{\Pi_0}(I_0) < g'(k)$.*

We now prove the claim by constructing an algorithm $\mathbb{B}$ that performs the following steps. Given I_0 and k, $\mathbb{B}$ applies the reduction on the instance I_0 with the parameters k and $r = \frac{f(k)}{h(f(k))}$. Denote by I_1 the instance of Π_1 produced by the reduction, so we have that $|I_1| = O_k(|I_0|^{O(1)})$. The following properties are immediate from the definitions of the FPT gap reductions (Definition 3.4).

- If $\mathsf{OPT}_{\Pi_0}(I_0) \ge k$, then we have $\mathsf{OPT}_{\Pi_1}(I_1) \ge f(k)$.
- If $\mathsf{OPT}_{\Pi_0}(I_0) < g'(k) := g(\frac{f(k)}{h(f(k))})$, then we have $\mathsf{OPT}_{\Pi_1}(I_1) < r = \frac{f(k)}{h(f(k))}$

Since $\mathbb{A}$ is an h-FPT gap approximation algorithm, running $\mathbb{A}$ on $(I_1, f(k))$ can distinguish between the above two cases. Consequently, ones can also invoke $\mathbb{A}$ to distinguish between the cases that $\mathsf{OPT}_{\Pi_0}(I_0) \ge k$ and that $\mathsf{OPT}_{\Pi_0}(I_0) < g'(k) = g(\frac{f(k)}{h(f(k))})$ in time $O_k(|I_1|^F) = O_k(|I_0|^{DF}) = O_k(|I_0|^{O(1)})$. Notice also that

$$g'(k) = g(\frac{f(k)}{h(f(k))}) \le g(f(k)) \le k$$

where the first inequality is because $f(k)/h(f(k)) \le f(k)$ (recall that $h(f(k)) \ge 1$ by Definition 2.1) and because g is non-decreasing, and the second inequality is by the claim below.

Claim B.2. *For any totally-FPT-inapproximable problem Π_0, any functions g and f that satisfy conditions in Definition 3.4 and any integer x, it holds that $g(f(x)) \leq x$.*

Proof. For any integer x, consider an instance I_0 such that $\mathsf{OPT}_{\Pi_0}(I_0) \geq x$ (such I_0 exists because $\mathsf{OPT}_{\Pi_0} = \omega(1)$; otherwise, Π_0 is not totally-FPT-inapproximable (e.g., we can always output 1 if Π_0 is a maximization problem)). By the second condition in Definition 3.4, $\mathsf{OPT}_{\Pi_1}(I_1) \geq f(x)$. Applying the contrapositive of the third condition with $r = f(x)$ (thus, $\mathsf{OPT}_{\Pi_1}(I_1) \geq r$), we have $\mathsf{OPT}_{\Pi_0}(I_0) \geq g(r) = g(f(x))$. Thus, $x \geq \mathsf{OPT}_{\Pi_0}(I_0) \geq g(f(x))$ as claimed. □

To complete the proof, one only needs to argue that $g(\frac{f(k)}{h(f(k))}) = \omega(1)$, and this simply follows from the fact that $f(k) = \omega(1)$, $g(k) = \omega(1)$ and that $k/h(k) = \omega(1)$.

C FTP-Inapproximability under W[1]-Hardness

In this section, we show an example of problems that have no FPT-approximation algorithm unless W[1]=FPT, namely the *maximum induced path* problem (InducedPath).

In InducedPath, we are given a graph G, and the goal is to find a maximum size subset of vertices $S \subseteq V(G)$ such that S induces a path in G. We will show that InducedPath has no FPT-approximation algorithm. Implicit in our reduction is a reduction from k-Clique to the *multi-colored clique* problem.

Theorem C.1. *Unless W[1]=FPT, for any positive integers $q : 1 \leq q \leq n^{1-\delta}$, for any $\delta < 0$, given a graph G on n vertices and for any function $t : \mathbb{R} \to \mathbb{R}$, there is no $t(k)\,\mathrm{poly}(n)$-time algorithm that distinguishes between the following two cases:*

- $\mathsf{InducedPath}(G) \geq 2q \cdot k$.
- $\mathsf{InducedPath}(G) \leq 4(k-1)$.

Proof. The reduction is as follows. Take a graph H of a k-Clique instance. Then we construct a graph G as follows. First, we create intermediate graphs $Z_1, \ldots, Z_q$. Each graph Z_i for $i \in [q]$ is created by making k copies of $V(H)$, namely, $V_{i,1}, \ldots, V_{i,k}$ and forming a clique on $V_{i,j}$ for each $j \in [k]$. So, now, we have k disjoint cliques. For each vertex $v \in V(H)$, we pick a copy of v, one from each $V_{i,j}$, say $v_{i,j}$, and we form a clique on $\{v_{i,1}, \ldots, v_{i,k}\}$. Next, for each edge $uv \notin E(H)$, we add edges $u_{i,j}v_{i,j'}$, for all $j, j' \in [k]$, where $u_{i,j}$ and $v_{i,j'}$ are the copy of u in $V_{i,j}$ and the copy of v in $V_{i,j'}$, respectively. Next, we add a dummy vertex $x_{i,j}$ for each $V_{i,j}$ and add edges joining $x_{i,j}$ to every vertex of $V_{i,j}$ and to every vertex of $V_{i,j-1}$ if $j \geq 2$. Finally, we join the graph Z_i for all $i \in [q]$ to be of the form $(Z_1, Z_2, \ldots, Z_k)$. To be precise, for each graph Z_i with $i \geq 2$, we join the vertex $x_{i,1}$ (which belongs to Z_i) to every vertex of $V_{i-1,q}$ (which belongs to Z_{i+1}).

Completeness. First, suppose that $\mathsf{Clique}(H) \geq k$. We will show that $\mathsf{InducedPath}(G) \geq 2q \cdot k$. We take a subset of vertices $S \subseteq V(H)$ that induces a clique on H. Let us name vertices in S by $v^1, \ldots, v^k$. For each $j \in [k]$, we pick the copies $v^j_{i,j}$ of v^j from $V_{i,j}$ for all $i \in [q]$. We then pick all the vertices $x_{i,j}$ for $i \in [k]$ and $j \in [q]$. We denote this set of vertices by S'. It is not hard to see that for any distinct vertices $v^j, v^{j'} \in S$, their copies $v^j_{i,j}$ and $v^{j'}_{i',j'}$ are not adjacent, and each vertex $x_{i,j}$ has exactly two neighbors: $v^j_{i,j}$ and $u^{j-1}_{i,j-1}$ (or $u^k_{i-1,k}$). Therefore, S' induces a path in G of size $2qk$.

Soundness. Suppose that $\mathsf{Clique}(H) < k$, i.e., H has no clique of size k. We will show that $\mathsf{InducedPath}(G) \leq 4(k-1)$. To see this, let $S' \subseteq V(G)$ be a subset of vertices that induces a path $G[S']$ in G. Observe that, for $i \in [q]$, $G[S'] \cap Z_i$ must be a path of the form $(x_{i,a}, v^a_{i,a}, \ldots, x_{i,k}, v^b_{i,b})$. Moreover, $v^\ell_{i,\ell}$ and $v^{\ell'}_{i,\ell'}$ are not adjacent in G for any $\ell \neq \ell'$, meaning that $v^\ell_{i,\ell}$ and $v^{\ell'}_{i,\ell'}$ are not copies of the same vertex in H, while the set $\{v^\ell\}_{a \leq \ell \leq b}$ induces a clique in H. Thus, $a - b + 1 < k$, and $G[S'] \cap Z_i$ can have at most $2(k-1)$ vertices. It follows that any induced path $G[S']$ of G can contain vertices from at most two subgraphs, say Z_i and Z_{i+1}. Therefore, we conclude that $|S'| \leq 4(k-1)$. □

The FPT-inapproximable of InducedPath follows directly from Theorem C.1.

Corollary C.2. *Unless W[1]=FPT, there is no $f(k)$-approximation algorithm for* InducedPath *that runs in $t(k)\,\mathrm{poly}(n)$-time for any functions f and t depending only on k.*

D Known Connections between Problems

In this section, we discuss known equivalences between problems in more detail.

Dominating Set and Set Cover: It is easy to see that DomSet is a special case of SetCov, and the reduction from SetCov to DomSet is by phrasing $\mathcal{U}$ and $\mathcal{S}$ as vertices, forming a clique on $\mathcal{S}$ and there is an edge joining a subset $S_i \in \mathcal{S}$ and element $u_j \in \mathcal{U}$ if and only if u_j is an element in S_i.

Induced Matching and Independent Set: We show that Induced Matching is at least as hard to approximate as Independent Set. Let G be an input graph of Independent Set. We create a graph G' by, for each vertex $v \in V(G)$, create a vertex v' and an edge vv'. Notice that any independent set S of G corresponds to an induced matching in G': For each $v \in S$, we have an edge vv' in the set $\mathcal{M}$. Conversely, for any induced matching $\mathcal{M}$ of G', we may assume that the matching only chooses edges of the form vv'.

More hereditary properties: We discuss some more natural problems in this class. If we define Π to be a set of all planar graphs, this is hereditary. The corresponding optimization problem is that of computing a maximum induced planar graphs. If we define Π to be a set of all forests, this is also hereditary, and it gives the problem of computing a maximum induced forest.

E Proof Sketch of Theorem 4.1

We will sketch the proof of Theorem 4.1.

In the forward direction, we use a standard reduction, which is sometimes referred to as the *clause-variable game* [AIM14]. Specifically, we transform a 3-SAT instance ψ on n variables $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$ into a label cover instance $\Gamma = (G = (U, V, E), \Sigma_U, \Sigma_V, \Pi)$ by transforming clauses into left vertices in U and variables into right vertices in V, and there is an edge joining a pair of vertices C_i and x_j if x_j appears in C_i. We take partial assignments as the label sets Σ_U and Σ_V, and a constraint on each edge asks for a pair (α, β) of labels that are *consistent*, i.e., they assign the same value to the same variable (e.g., $\alpha = (x_1 : 1, x_2 : 0, x_3 : 1)$ and $\beta = (x_1 : 1)$ are consistent whereas α is not consistent with $\beta' = (x_2 : 1)$), and α causes C_i to evaluate to true (i.e., some of the literal in C_i is assigned to true by α). We denote the evaluation of a clause C_i on a partial assignment α by $C_i(\alpha)$.

To be precise, we have

$$U = \{C_1, \ldots, C_m\}, \quad V = \{x_1, \ldots, x_n\}, \quad E = \{C_i x_j : x_j \text{ appears in the clause } C_i\}$$
$$\Sigma_U = \{0,1\}^3, \quad \Sigma_V = \{0,1\}, \quad \Pi_{C_i x_j} = \{(\alpha, \beta) : \alpha \text{ and } \beta \text{ are consistent} \wedge C_i(\alpha) = \text{true}\}$$

It can be seen that $\mathsf{MaxCov}(\Gamma) = \mathsf{SAT}(\psi)$ since the only way to cover each node $C_i \in U$ is to pick assignments to all vertices adjacent to C_i so that they are all consistent with the assignment $\alpha = \sigma_V(C_i)$ (and that $C_i(\alpha) = \text{true}$).

The converse direction is not straightforward. We apply Håstad [Hås01] reduction[10] to reduce an instance Γ of MaxCov to a 3-SAT instance of size $f(|\Sigma_U| + |\Sigma_V|) \cdot O(|U| + |V|)$ with a hardness gap $1-\varepsilon$, for some constant $\varepsilon > 0$ (the hardness gap is different from the original MaxCov instance). Note that f in the Håstad's construction is a doubly exponential function. The equivalent between MaxCov and 3-SAT holds only when $|\Sigma_U| + |\Sigma_V|$ is constant (or at most $\log\log(|V| + |U|)$). □

F On Gap-ETH

While Gap-ETH may sound like a very strong assumption, as pointed out in [Din16; MR16], there are a few evidences suggesting that the conjecture may indeed be true:

- In a simplified and slightly inaccurate manner, the PCP theorem [AS98; ALMSS98] can be viewed as a polynomial time reduction that takes a 3-CNF formula Φ and then produces another 3-CNF formula Φ' such that, if Φ is satisfiable, then Φ' is satisfiable, and, if Φ is unsatisfiable, Φ' is not only unsatisfiable but also not even 0.99-satisfiable. To date, it is known that the size of Φ' can be made as small as $n \operatorname{polylog}(n)$ where n is the size of Φ [Din07]. This means that, assuming ETH, Gap-3SAT cannot be solved in $2^{o(n/\operatorname{polylog} n)}$ time, which is only a factor of $\operatorname{polylog} n$ away from what we need in Gap-ETH. Indeed, as stated earlier, if a linear-size PCP exists (which implies that the size of Φ' can be made linear in n), then Gap-ETH would follow from ETH.

- No subexponential-time algorithm is known even for the following (easier) problem, which is sometimes referred to as *refutation of random 3-SAT*: for a constant density parameter Δ, given a 3-CNF formula Φ with n variables and $m = \Delta n$ clauses, devise an algorithm that outputs either SAT or UNSAT such that the following two conditions are satisfied:

  - If Φ is satisfiable, then the algorithm always output SAT.
  - Over all possible 3-CNF formulae Φ with n clauses and m variables, the algorithm outputs UNSAT on at least 0.5 fraction of them.

  Note that, when Δ is a sufficiently large constant (say 1000), a random 3-CNF formula is, with high probability, not only unsatisfiable but also not even 0.9-satisfiable. Hence, if Gap-ETH fails, then the algorithm that refutes Gap-ETH will also be a subexponential time algorithm for refutation of random 3-SAT with density Δ.

  Refutation of random 3-SAT, and more generally random CSPs, is an important question that has connections to many other fields, including hardness of approximation, proof complexity,

[10] Here we apply only the Hastad's reduction from label cover to 3SAT, without parallel repetition.

cryptography and learning theory. We refer the readers to [AOW15] for a more comprehensive survey on the problem and its applications in various areas. Despite being intensively studied for almost three decades, no subexponential-time algorithm is known for the above regime of parameter. In fact, it is known that the Sum-of-Squares hierarchies cannot refute random 3-SAT with constant density in subexponential time [Gri01; Sch08]. Given how powerful SDP [Rag08], and more specifically Sum-of-Squares [LRS15], are for solving (and approximating) CSPs, this suggests that refutation of random 3-SAT with constant density, and hence Gap-3SAT, may indeed be exponentially hard or, at the very least, beyond our current techniques.

- Dinur speculated that Gap-ETH might follow as a consequence of some cryptographic assumption [Din16]. This was recently confirmed by Applebaum [App17] who showed that Gap-ETH follows from an existence of any exponentially-hard locally-computable one-way function. In fact, he proved an even stronger result that Gap-ETH follows from ETH for some CSPs that satisfy certain "smoothness" properties.

Lastly, we note that the assumption $m = O(n)$ made in the conjecture can be made without loss of generality. As pointed out in both [Din16] and [MR16], this follows from the fact that, given a 3-SAT formula ϕ with m clauses and n variables, if we create another 3-SAT formula ϕ' by randomly selected $m' = \Delta n$ clauses, then, with high probability, $|\mathsf{SAT}(\phi)/m - \mathsf{SAT}(\phi')/m'| \leq O(1/\Delta)$.