58th Annual IEEE Symposium on Foundations of Computer Science

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[‖††]
[*]*CS, Aalto University, Espoo, Finland. Email:* parinya.chalermsook@aalto.fi
[*]*Informatics Institute, University of Warsaw, Warsaw, Poland. Email:* cygan@mimuw.edu.pl
[‡]*CS, Rutgers University-Camden, Camden, NJ, USA. Email:*guyk@camden.rutgers.edu
[§]*Math & CS, Weizmann Institute of Science, Rehovot, Israel. Email:* bundit.laekhanukit@mail.mcgill.ca
[¶]*ITCS, Shanghai University of Finance and Economics, Shanghai, China.*
[‖]*EECS, UC Berkeley, Berkeley, CA, USA. Email:* {pasin, luca}@berkeley.edu
[**]*CSC, KTH Royal Institute of Technology, Stockholm, Sweden. Email:* danupon@kth.se
[††]*Simons Institute for the Theory of Computing, UC Berkeley, Berkeley, CA, USA.*

***Abstract*—We consider questions that arise from the intersection between the areas of approximation algorithms, subexponential-time algorithms, and fixed-parameter tractable algorithms. The questions, which have been asked several times (e.g., [1], [2], [3]) 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) [4], [5], 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 Balanced 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 [6], [7], [8]. 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 [9], [10] and $\log^{1/4+\epsilon}(\mathsf{OPT})$ approximation for DomSet for any constant $\epsilon > 0$ [11]. Our result on Clique significantly improves on [9], [10]. However, our result on DomSet is incomparable to [11] since their results hold under ETH while our results hold under Gap-ETH, which is a stronger assumption.

***Keywords*-Fixed Parameter Tractability; Hardness of Approximation; Clique; Set Cover; Dominating Set**

I. 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, 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 it outputs an *approximation* of the optimal solution; however, the approximation factor is analyzed in terms of the optimum (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* when $f(\mathsf{OPT})$ is $o(\mathsf{OPT})$, while for a minimization problem, it is non-trivial for any computable function f.

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 $O(\frac{n(\log\log n)^2}{\log^3 n})$ with clever ideas [12]. Observe, however, that this bound are quite meaningless when $\mathsf{OPT} = O(\frac{n(\log\log n)^2}{\log^3 n})$ since outputting a single vertex already guarantees such bound. In this case, a bound such as $O(\frac{\mathsf{OPT}}{\log\log\mathsf{OPT}})$ would be more meaningful.

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

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 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., [1], [2], [3]). 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. [9] and Bonnet et al. [10], assuming the Exponential Time Hypothesis (ETH) and that a linear-size PCP exists. Alternatively, Khot and Shinkar [15] proved this under an assumption that solving quadratic equations over a finite field under a certain regime of parameters is not in FPT; alas, this assumption was later shown to be false [16]. For DomSet, Chen and Li [11] could rule out $O(1)$-FPT-approximation algorithms assuming $\mathsf{FPT} \neq \mathsf{W}[1]$. Moreover, they improved the inapproximability ratio to $\log^{1/4+\epsilon}(\mathsf{OPT})$ for any constant $\epsilon > 0$ under ETH. Note that ETH implies that $\mathsf{FPT} \neq \mathsf{W}[1]$.

Our Results and Techniques. We show that there is no non-trivial FPT-approximation algorithm for 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 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 II).

Gap-ETH, first formalized in [4], [5], is a stronger version of the aforementioned ETH, which only asserts that no subexponential time algorithms can decide whether a given 3SAT formula is satisfiable. It has recently been shown to be useful in proving fine-grained hardness of approximation for problems such as dense CSP with large alphabets [5] and Densest-k-Subgraph with perfect completeness [17].

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 [9], [18] under the same (in fact, weaker) assumption. Our result for DomSet also improves upon the results in [11] in terms of inapproximability ratio, 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 [19], [20]. 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 [21] or the randomized graph product [22]), while amplifying the gap to arbitrarily large, cause the input size to be too large that existing OPT reduction techniques (e.g., [19], [23]) 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. 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 [22] and, in fact, the reduction itself has been studied before [24], [25] but in a very different regime of parameters. For a more detailed discussion, please refer to Subsection IV-B.

[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 $f' = o(\mathsf{OPT})$ [13], [1] (also see [14]). So, it does not matter whether the running time is polynomial on the size of the input or depends on OPT.

[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 IV for more detail.

Once the inapproximability results for label cover problems with small left-hand-side and right-hand-side vertex set are established, we can reduce it to Clique and DomSet using the standard reductions from [26] and [27] respectively.

Besides the results for Clique and DomSet, we also show that no non-trivial FPT-approximation algorithm exists for a few other problems, including *Maximum Biclique* (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 [6], [7], [8]. Additionally, we rule out $k^{o(1)}$-FPT-approximation algorithms for *Densest k-Subgraph* (DkS) although this ratio does not yet match the trivial $O(k)$-approximation algorithm. 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 DkS in [17].

Due to space constraint, we will only focus on Clique and DomSet here; for the other results, please refer to our full version, which is available online as [28].

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 et al. in [29] (the journal version appears in [19]). 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, Pătraşcu and Williams [23] 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 [9] and by Bonnet, Lampis and Paschos in [10]. Our techniques can be seen as introducing gap amplification to the reductions in [19]. We emphasize that while [19],[23],[9] and [10] (and also the reductions in this paper) are all based on the technique of compressing the optimal solution, Hajiaghayi et al. [9] compress the optimal solution after reducing SAT to the designated problems, i.e., Clique and DomSet. The reductions in [19], [23], [10] and this 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 [9] 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 [9].

Recently, Chen and Lin [11] showed that DomSet admits no constant approximation algorithm unless $\mathrm{FPT} = \mathrm{W}[1]$. Their hardness result was derived from the seminal result of Lin [6], 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 [6] and the proof of the ETH-hardness of Clique [29]. 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 [29].

Our method for proving the FPT inapproximability of DomSet is similar to that in [23]. However, the original construction in [23] does not require a "partition system". This is because Pătraşcu and Williams reduction starts from SAT, which can be casted as DomSet. In our construction, the reduction starts from a label cover problem 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 [30], [27].)

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 [11]. If ones introduced Gap-ETH to the previous works, then the proofs in [19], [9], [10] yield the constant FPT-inapproximability of Clique, and the proof in [19] yields the constant FPT-inapproximability of DomSet.

The summaries of previous works are presented in Table I.

II. Preliminaries

For any graph G, we denote by $V(G)$ and $E(G)$ the vertex and edge sets of G, respectively. For each $u \in V(G)$, we denote the set of its neighbors by $N_G(v)$. A *clique* of G is a complete subgraph of G. The *clique number* of G, denoted by $\mathsf{Clique}(G)$, is the size of the largest clique in G. A *dominating set* of G is a subset $S \subseteq V(G)$ such that every vertex in G is either in S or has a neighbor in S.

A. FPT Approximation

We employ the following notations of optimization problems from [33]. An *optimization problem* Π is defined by three components: (1) for each 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

[4]Constant FPT-inapproximability of Clique under ETH is claimed in [9] (arXiv version). However, as we investigated, Gap-ETH is assumed there.

Summary of Works on Clique

Inapprox	Running Time	Assumption	Ref
any constant	$t(OPT) \cdot n^{o(\mathsf{OPT})}$	ETH + LPCP	[10]
$\mathsf{OPT}^{1-\epsilon}$	$\exp(\mathsf{OPT}^{\rho(\epsilon)})$	ETH	[31]
$1/(1-\epsilon)$	$\exp(\exp(\mathsf{OPT}^{\rho(\epsilon)}))^4$	ETH	[9]
$o(\mathsf{OPT})$	$t(OPT) \cdot n^{o(\mathsf{OPT})}$	Gap-ETH	*

Summary of Works on DomSet

Inapprox	Running Time	Assumption	Ref
$\mathsf{OPT}^{1-\gamma}$	$\exp(\mathsf{OPT}^{1-\rho(\gamma)})$	ETH	[31]
$(\log \mathsf{OPT})^{\delta}$	$\exp(\exp((\log \mathsf{OPT})^{\delta-1}))$	ETH + PGC	[9]
any constant	$t(OPT) \cdot n^{O(1)}$ (i.e. no FPT)	$\mathrm{W}[1] \neq \mathrm{FPT}$	[11]
$(\log \mathsf{OPT})^{1/4+\epsilon}$	$t(OPT) \cdot n^{o(\sqrt{\mathsf{OPT}})}$	ETH	[11]
$f(\mathsf{OPT})$	$t(OPT) \cdot n^{o(\mathsf{OPT})}$	Gap-ETH	*

Table I: Summaries of previous works. Here t denotes any computable function t, 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 stand for the *Projection Game Conjecture* [32] and *Linear-Size PCP Conjecture* [10] respectively. Stars in reference columns denote our works.

$\mathsf{GOAL}_\Pi \in \{\min, \max\}$ 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 additional inputs k, f 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 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 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. *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 $f(k) = o(k)$, Π is not f-FPT gap approximable.*

Both Clique and DomSet 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 as discussed in the introduction. These implications are formalized in the two propositions below; their proofs are deferred to Appendix A of the full version [28].

Proposition 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 y such that $\mathsf{COST}_\Pi(I, y) \leq \mathsf{OPT}_\Pi(I) \cdot f(\mathsf{OPT}_\Pi(I))$.

Proposition 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 y such that $\mathsf{COST}_\Pi(I, y) \geq \mathsf{OPT}_\Pi(I)/f(\mathsf{OPT}_\Pi(I))$.

B. List of Problems

While both Clique and DomSet can be defined in terms of optimization problems similar to the previous subsection, we will omit the terms SOL, COST and GOAL since 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 size of the maximum clique in 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.

C. 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. For a formula ϕ, let $\mathsf{SAT}(\phi)$ denote the maximum number of clauses satisfied by any assignment.

Gap-ETH can now be stated in terms of SAT as follows.

Conjecture 5 ((randomized) Gap Exponential-Time Hypothesis (Gap-ETH) [4], [5])**.** *For some constant $\delta, \epsilon > 0$, no algorithm can, given a 3-SAT formula ϕ on n variables and $m = O(n)$ clauses, distinguish 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 [34]. 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. Some of these evidences are discussed in Appendix F of the full version of this paper.

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; please see Section 4.2.1 of the full version for more details.

III. 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]. This running time is nearly the best possible assuming ETH: Any algorithm that finds a k-clique in time $|V(G)|^{o(k)}$ would refute 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 6 (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 $\mathsf{Biclique}$. Even for the exact version of $\mathsf{Biclique}$, the best running time lower bound known is only $|V(G)|^{\Omega(\sqrt{k})}$ [6] assuming ETH. In order to succinctly categorize such lower bounds, we define a similar but weaker notation of *weakly* inherently enumerative:

Definition 7 (Weakly Inherently Enumerative)**.** *For any function $\beta = \omega(1)$, 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 $\beta = \omega(1)$.*

As stated earlier, we will prove total FPT inapproximability through inherently enumerative; the proposition below establishes a connection between the two.

Proposition 8. *If Π is weakly inherently enumerative, then Π is totally FPT inapproximable.*

For the purpose of facilitating proofs of totally FPT inapproximability, we define the following reduction, which we call *FPT gap reductions*.

Definition 9 (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 in an instance I_0 of Π_0 and integers q, r and produce 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 $t : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$.*
- *If* $\mathsf{OPT}_{\Pi_0}(I_0) \geq q$*, then* $\mathsf{OPT}_{\Pi_1}(I_1) \geq f(q)$.
- *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 total FPT inapproximability and weakly inherently

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

enumerative, as formalized below. Due to space constraint, we defer all proofs in this section to the full version [28].

Proposition 10. *If Π_0 is totally FPT inapproximable and is (f,g)-FPT gap reducible to Π_1 for some computable non-decreasing $f, g = \omega(1)$, Π_1 is totally FPT inapproximable.*

Proposition 11. *If Π_0 is β-weakly inherently enumerative and is (f,g)-FPT gap reducible to Π_1 for some $f, g, \beta = \omega(1)$, then Π_1 is $\Omega(\beta \circ g)$-weakly inherently enumerative.*

IV. 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.

A. 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 with vertex sets U, 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 $u \in U, v \in V$) 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.,

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

The goal of the Max-Cover problem is to compute $\mathsf{MaxCov}(\Gamma)$. We remark that the standard label cover problem (e.g., [35]) 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 cover every edge in G.

Note that we can assign multiple labels to vertices in V while each vertex in U must be assigned a unique label. This makes MinLab different from the problem known in the literature as MinRep (e.g., [35]) since MinRep allows us to assign multiple labels to all vertices, even those in U.

Results. First, note that checking whether $\mathsf{MaxCov}(\Gamma) < r$, 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 $\mathsf{MaxCov}(\Gamma) \leq (1-\varepsilon)|U|$:

Theorem 12. *Gap-ETH (Conjecture 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 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 the theorem is standard and is therefore omitted from this extended abstract.

We will show that Theorem 12 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. 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 13 to 15 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.

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

Theorem 13 (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; in particular when $|V| \ll |U|$. In this case, we cannot make the soundness (i.e., the parameter r in Theorem 13) 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 need later.

Theorem 14 (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 12 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 15 (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 the proofs of Theorems 13 to 15.

B. *Proof of Theorem 13*

The proof proceeds by **compressing the left vertex set** U of a label cover instance from Theorem 12. 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., [36]), it is more convenient for us to describe it in terms of subsets.

Definition 16. *For any positive integers $m, k, \ell, r \in \mathbb{N}$ and any constant $\varepsilon \in (0,1)$, an (m,k,ℓ,r,ε)-*disperser *is a collection $\mathcal{I}$ of k subsets $I_1, \dots, I_k \subseteq [m]$ each of size ℓ such that the union of any r different subsets from the collection has size at least $(1-\varepsilon)m$.*

The idea of using dispersers to amplify gap in hardness of approximation bears a strong resemblance to the randomized graph product [22]. Indeed, similar approaches have been used before, both implicitly (e.g., [37]) and explicitly (e.g., [24], [25], [38]). In fact, even the reduction we use below has been studied before by Zuckerman [24], [25]!

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 only aim to prove a running time lower bound of $|\Sigma_U|^{\Omega(r)}$.

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 13.

Claim 17. *For any $m, k, r \in \mathbb{N}$ and any $\varepsilon \in (0,1)$, let $\ell = \max\{m, \lceil 3m/(\varepsilon r)\rceil\}$ and let $I_1, \dots, I_k$ be ℓ-element subsets of $[m]$ drawn uniformly independently at random. If $\ln k \leq m/r$, then $\mathcal{I} = \{I_1, \dots, I_k\}$ is an (m,k,ℓ,r,ε)-disperser with probability at least $1 - e^{-m}$.*

The proof of Claim 17 is via standard probabilistic method and is deferred to the full version. With the definition of dispersers and Claim 17 ready, we now prove Theorem 13.

Proof of Theorem 13: 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 12. 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.

Recall the parameter ε from Theorem 12 and the parameters k, r from the statement of Theorem 13. Let $\ell := 3m/(\varepsilon r)$ 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 are k vertices in U where each vertex is a random set of ℓ vertices of $\widetilde{U}$. More specifically, we define $U = \{I_1, \dots, I_k\}$ 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**; to satisfy I_i, the assignment must satisfy all clauses in I_i.

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

Completeness. Suppose there is a labeling $(\sigma_{\widetilde{U}}, \sigma_{\widetilde{V}})$ of $\widetilde{\Gamma}$ that covers all vertices in $\widetilde{U}$. 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, $\text{MaxCov}(\Gamma) = |U|$.

Soundness. Recall Claim 17 that $\{I_1, \dots, I_k\}$ is an $(m, k, \ell, r, \varepsilon)$-disperser w.h.p. Conditioned on this event happening, we will prove the soundness property, i.e., if $\text{MaxCov}(\widetilde{\Gamma}) < (1 - \varepsilon)|\widetilde{U}|$, then $\text{MaxCov}(\Gamma) < r$.

We prove by contrapositive. Assume that there is a labeling (σ_U, σ_V) that covers at least r vertices $I_{i_1}, \cdots, I_{i_r} \in U$. We construct a labeling $(\sigma_{\widetilde{U}}, \sigma_{\widetilde{V}})$ as follows. First, let $\sigma_{\widetilde{V}} = \sigma_V$. 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, just 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, $\text{MaxCov}(\widetilde{\Gamma}) \geq (1 - \varepsilon)|\widetilde{U}|$ as desired.

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 12. 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 $\text{MaxCov}(\Gamma) = k$ or $\text{MaxCov}(\Gamma) < r$ 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 $\text{MaxCov}(\widetilde{\Gamma}) = |\widetilde{U}|$ or $\text{MaxCov}(\widetilde{\Gamma}) < (1 - \varepsilon)|\widetilde{U}|$ in time $O(2^{\delta_0 m})$. From Theorem 12, this is indeed a contradiction. ∎

C. *Proof of Theorem 14*

The proof proceeds by **compressing the right vertex set** V of a label cover instance from Theorem 12 plus amplifying the hardness gap. The gap amplification step is similar to that in the proof of Theorem 13 except that, since here $\text{MaxCov}(\Gamma)$ 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.

Proof of Theorem 14: 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 12. We may assume that $|\Sigma_{\widetilde{U}}|, |\Sigma_{\widetilde{V}}| = O(1)$, and $|\widetilde{U}| = \Theta(|\widetilde{V}|)$. We also assume w.l.o.g. that $\widetilde{U} = [m]$ and $\widetilde{V} = [n]$.

Recall the parameter ε from Theorem 12 and the parameters q, γ from Theorem 14. 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, partition $\widetilde{V} = [n]$ into q parts $J_1, \dots, J_q$, each of size n/q. Then, let $V = \{J_1, \dots, J_q\}$. 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, $\Pi_{IJ} = \{(\alpha, \beta) = ((\alpha_u)_{u\in I}, (\beta_v)_{v\in J}) : \forall u \in I, v \in J$ 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 $I_1, \dots, I_t \in U$ be the vertices covered by (σ_U, σ_V). Analogous to the proof of Theorem 13, 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 $u \in I_{i_1} \cup \dots \cup I_{i_r}$, 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 $u \in U \setminus (I_{i_1} \cup \dots \cup I_{i_r})$, we set $\sigma_{\widetilde{U}}(u)$ arbitrarily.

Similar to the proof of Theorem 13, it is not hard to see that every vertex in $I_1 \cup \dots \cup I_t$ 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 $|I_1 \cup \dots \cup I_t| < (1-\varepsilon)|\widetilde{U}|$. Since each I_i is simply an ℓ-size subset of $I_1 \cup \dots \cup I_t$, we can conclude that

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

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

Running Time Lower Bound. Our construction gives an instance Γ with $|V| = q$ and $|\Sigma_V| = |\Sigma_{\widetilde{V}}|^{n/q} = 2^{\Theta(n/q)}$; moreover, $|U| = m^\ell$ 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 12. 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 12, this is a contradiction. ■

V. Proof of Theorem 15

The proof proceeds simply by showing that, if an algorithm can distinguish between the two cases in the statement of Theorem 15, it can also distinguish between the two cases in Theorem 14 (with an appropriate value of γ).

Proof of Theorem 15: Consider the label cover instance $\Gamma = (G = (U, V, E), \Sigma_U, \Sigma_V, \Pi)$ given by Theorem 14 with $\gamma = (r/q)^{-q}$. Assume w.l.o.g. that no vertex in G is isolated.

Completeness. If $\mathsf{MaxCov}(\Gamma) = |U|$, then there is a labeling (σ_U, σ_V) that covers every edge; this also induces a multi-labeling that covers every edge. Hence, $\mathsf{MinLab}(\Gamma) = |V|$.

Soundness. Suppose contrapositively that $\mathsf{MinLab}(\Gamma) \leq r$, i.e., there exists a multi-labeling $(\sigma_U, \hat{\sigma}_V)$ such that $\sum_{v\in V} |\hat{\sigma}_V(v)| \leq r$ and every vertex is covered. Since there is no isolated vertex in G, $\hat{\sigma}_V(v) \neq \emptyset$ for all $v \in V$.

Consider $\sigma_V : V \to \Sigma_V$ sampled randomly by, for each $v \in V$, letting $\sigma_V(v)$ be a random element of $\hat{\sigma}_V(v)$. From linearity of expectation, the expected number of $u \in U$ that are covered by the labeling (σ_U, σ_V) is equal to

$$\begin{aligned}\sum_{u\in U} \Pr_{\sigma_V}[(\sigma_U, \sigma_V) \text{ covers } u] &\geq \sum_{u\in U} \prod_{v\in N(u)} |\hat{\sigma}_V(v)|^{-1} \\ \text{(From AM-GM inequality)} &\geq \sum_{u\in U} \left(\frac{1}{q}\sum_{v\in V} |\hat{\sigma}_V(v)|\right)^{-q} \\ &\geq \sum_{u\in U} |U|(r/q)^{-q} = \gamma|U|.\end{aligned}$$

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

VI. Hardness for Combinatorial Problems

A. Maximum Clique

Recall that, for any graph G, we can decide whether $\mathsf{Clique}(G) \geq r$ by enumerating all r-vertex subsets of $V(G)$, which takes $|V(G)|^{O(r)}$ time. 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 18. *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 13.

Theorem 19 ([26]). *Given a label cover instance $\Gamma = (G = (U, V, E), \Sigma_U, \Sigma_V, \Pi)$ with projection property as in Section IV, 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, in our terminologies, the FGLSS graph [26] is 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 18: Assume that Gap-ETH holds and let δ, ρ be as in Theorem 13. 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 distinguish whether $\mathsf{MaxCov}(\Gamma) \geq q$ or $\mathsf{MaxCov}(\Gamma) < r$ as follows. We first run the FGLSS reduction to produce a graph H_Γ and 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 $O_{q,r}(|V(H_\Gamma)|^{\delta r}) + O(|V(H_\Gamma))|^2|V|) \leq O_{q,r}(|\Gamma|^{\delta r})$. From Theorem 13, this is a contradiction. ∎

Theorem 18 immediately implies the totally FPT inapproximability of Clique:

Corollary 20. *Assuming Gap-ETH, Maximum Clique is inherently enumerative and thus totally FPT inapproximable.*

B. Set Cover and Dominating Set

It is well-known that DomSet is equivalent to the *minimum set cover* problem (SetCov): Given a universe $\mathcal{U}$ of n elements and a collection $\mathcal{S}$ of m subsets $S_1, \dots, S_m \subseteq \mathcal{U}$, the goal is to find the minimum number of subsets of $\mathcal{S}$ whose union equals $\mathcal{U}$. We denote such number by $\mathsf{SetCov}(\mathcal{U}, \mathcal{S})$. Since it is more convenient for us, we will work with SetCov instead of DomSet.

Note that checking whether $\mathsf{SetCov}(\mathcal{U}, \mathcal{S}) \leq q$ can be done in $O^\star(|\mathcal{S}|^q)$ time by enumerating all q-element subsets of $\mathcal{S}$. We show that this essentially the best we can do: Even when the algorithm is promised the existence of a set cover of size q, it cannot find a set cover of size r for any constant r (even when $r \gg q$) in time $O_{q,r}((|\mathcal{S}||\mathcal{U}|)^{\delta q})$ for some constant $\delta > 0$ independent of q, r:

Theorem 21. *Assuming Gap-ETH, there exist constants $\delta, q_0 > 0$ such that, for any integers $r \geq q \geq q_0$, no algorithm can take a SetCov instance $(\mathcal{U}, \mathcal{S})$, and distinguish between the following cases in $O_{q,r}((|\mathcal{S}||\mathcal{U}|)^{\delta q})$ time:*

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

The above theorem follows from plugging in the following reduction from MinLab to SetCov to Theorem 22.

Theorem 22. *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 explanation of this reduction to Section VI-B1. For now, let us turn to the proof of Theorem 21.

Proof of Theorem 21: Assume that Gap-ETH holds and let δ, ρ be as in Theorem 15. Let $q_0 = \max\{\rho, c/\delta\}$ where c is the constant such that the running time of the reduction in Theorem 22 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 distinguish whether $\mathsf{MinLab}(\Gamma) \leq q$ or $\mathsf{MinLab}(\Gamma) > r$ as follows. We first run the reduction from Theorem 22 to produce a SetCov instance $(\mathcal{U}, \mathcal{S})$ and 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 $O_{q,r}((|\mathcal{U}||\mathcal{S}|)^{\delta q}) + O((|\mathcal{U}||\mathcal{S}|)^c) \leq O_{q,r}(|\Gamma|^{\delta q})$. From Theorem 15, this is a contradiction. ∎

As a corollary of Theorem 21, we immediately arrive at FPT inapproximability of Set Cover.

Corollary 23. *Assuming Gap-ETH, Set Cover is inherently enumerative and thus totally FPT inapproximable.*

1) Proof of Theorem 22: Our construction is based on a standard hypercube set system, as used by Feige [27] in proving the hardness of the *k-Maximum Coverage* problem. We provide 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 $\mathcal{U} = [z]^k$, and the collection $\mathcal{S}$ of *canonical sets* $\{X_{i,a}\}_{i \in [z], a \in [k]}$ 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 24. *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 $X_{1,a}, X_{2,a}, \dots, X_{z,a} \in \mathcal{S}'$.*

The proof of Proposition 24 is straightforward and is omitted from this version of the paper.

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 $|\mathcal{S}| = |V||\Sigma_V|$ and $|\mathcal{U}| = |U||V|^{|\Sigma_U|}$. This completes the description of our instance.

Analysis: We argue that $\mathsf{MinLab}(\Gamma) = \mathsf{SetCov}(\mathcal{U}, \mathcal{S})$.

First, we will show that $\mathsf{MinLab}(\Gamma) \le \mathsf{SetCov}(\mathcal{U}, \mathcal{S})$. Let $(\sigma_U, \hat{\sigma}_V)$ be an optimal solution of MinLab on Γ. 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 $\sum_v |\hat{\sigma}(v)|$, the cost of $\mathsf{MinLab}(\Gamma)$. We argue that this is a feasible set cover: For each u, since u is covered by $(\sigma_U, \hat{\sigma}_V)$, there is a label $b_v \in \hat{\sigma}_V(v)$ such that $(\sigma_U(u), b_v) \in \Pi_{uv}$ for every $v \in N_G(u)$. 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 24. 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 24) 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 $(\sigma_U, \hat{\sigma}_V)$.

VII. Conclusion and Discussions

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 $\mathsf{W}[1]$ and $\mathsf{W}[2]$ respectively, it might be possible that these two problems can be sources of FPT-inapproximability of many other problems that admit no FPT algorithms.

Remark also 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 $\mathsf{FPT} \neq \mathsf{W}[2]$ in [39]. For the induced path problem, we show in Appendix C of the full version that it is FPT-inapproximable assuming $\mathsf{FPT} \neq \mathsf{W}[1]$. It would be interesting to understand whether it is possible to also base total FPT-inapproximability of Clique and DomSet under assumptions that are weaker than Gap-ETH, such as $\mathsf{FPT} \neq \mathsf{W}[1]$ or ETH. To this end, we note that Chen and Lin [11] showed the inapproximability of DomSet under $\mathsf{FPT} \neq \mathsf{W}[1]$ (resp., ETH), but their inapproximability ratio is only any constant (resp., $\log^{1/4-\varepsilon}(\mathsf{OPT})$); if their result could be extended to exclude $f(\mathsf{OPT})$-approximation for any function f, then DomSet would indeed be totally FPT-inapproximable under weaker assumptions.

Another interesting 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 [40]. 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 [16]. 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.

References

[1] D. Marx, “Parameterized complexity and approximation algorithms,” *Comput. J.*, vol. 51, no. 1, pp. 60–78, 2008.

[2] M. R. Fellows, J. Guo, D. Marx, and S. Saurabh, “Data Reduction and Problem Kernels (Dagstuhl Seminar 12241),” *Dagstuhl Reports*, vol. 2, no. 6, pp. 26–50, 2012.

[3] R. G. Downey and M. R. Fellows, *Fundamentals of Parameterized Complexity*. Springer, 2013.

[4] I. Dinur, “Mildly exponential reduction from gap 3SAT to polynomial-gap label-cover,” *ECCC*, vol. 23, p. 128, 2016.

[5] P. Manurangsi and P. Raghavendra, “A birthday repetition theorem and complexity of approximating dense csps,” *CoRR*, vol. abs/1607.02986, 2016.

[6] B. Lin, “The parameterized complexity of *k*-biclique,” in *SODA*, 2015, pp. 605–615.

[7] S. Khot and V. Raman, “Parameterized complexity of finding subgraphs with hereditary properties,” in *COCOON*, 2000, pp. 137–147.

[8] H. Moser and S. Sikdar, “The parameterized complexity of the induced matching problem,” *Discrete Applied Mathematics*, vol. 157, no. 4, pp. 715–727, 2009.

[9] M. T. Hajiaghayi, R. Khandekar, and G. Kortsarz, “Fixed parameter inapproximability for clique and set cover in time super-exponential in OPT,” *CoRR*, vol. abs/1310.2711, 2013.

[10] E. Bonnet, B. Escoffier, E. J. Kim, and V. T. Paschos, “On subexponential and FPT-time inapproximability,” *Algorithmica*, vol. 71, no. 3, pp. 541–565, 2015.

[11] Y. Chen and B. Lin, “The constant inapproximability of the parameterized dominating set problem,” in *FOCS*, 2016, pp. 505–514.

[12] U. Feige, “Approximating maximum clique by removing subgraphs,” *SIAM J. Discrete Math.*, vol. 18, no. 2, pp. 219–225, 2004.

[13] M. Grohe and M. Grüber, “Parameterized approximability of the disjoint cycle problem,” in *ICALP*, 2007.

[14] D. Marx, “Completely inapproximable monotone and antimonotone parameterized problems,” *J. Comput. Syst. Sci.*, vol. 79, no. 1, pp. 144–151, 2013.

[15] S. Khot and I. Shinkar, “On hardness of approximating the parameterized clique problem,” in *ITCS*, 2016, pp. 37–45.

[16] N. Kayal, “Solvability of systems of polynomial equations over finite fields,” 10 2014, a talk given by Neeraj Kayal at the Simons Institute for the Theory of Computing, Berkeley, CA [Accessed: 2017/20/7]. [Online]. Available: https://simons.berkeley.edu/talks/neeraj-kayal-2014-10-13

[17] P. Manurangsi, “Almost-polynomial ratio eth-hardness of approximating densest k-subgraph,” in *STOC*, 2017, pp. 954–961.

[18] É. Bonnet, M. Lampis, and V. T. Paschos, “Time-approximation trade-offs for inapproximable problems,” in *STACS*, 2016, pp. 22:1–22:14.

[19] J. Chen, X. Huang, I. A. Kanj, and G. Xia, “On the computational hardness based on linear FPT-reductions,” *J. Comb. Optim.*, vol. 11, no. 2, pp. 231–247, 2006.

[20] ——, “Strong computational lower bounds via parameterized complexity,” *J. Comput. Syst. Sci.*, vol. 72, no. 8, pp. 1346–1367, 2006.

[21] R. Raz, “A parallel repetition theorem,” *SIAM J. Comput.*, vol. 27, no. 3, pp. 763–803, 1998.

[22] P. Berman and G. Schnitger, “On the complexity of approximating the independent set problem,” *Inf. Comput.*, vol. 96, no. 1, pp. 77–94, 1992.

[23] M. Patrascu and R. Williams, “On the possibility of faster SAT algorithms,” in *SODA*, 2010, pp. 1065–1075.

[24] D. Zuckerman, “Simulating BPP using a general weak random source,” *Algorithmica*, vol. 16, no. 4/5, pp. 367–391, 1996.

[25] ——, “On unapproximable versions of np-complete problems,” *SIAM J. Comput.*, vol. 25, no. 6, pp. 1293–1304, 1996.

[26] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy, “Interactive proofs and the hardness of approximating cliques,” *J. ACM*, vol. 43, no. 2, pp. 268–292, 1996.

[27] U. Feige, “A threshold of ln *n* for approximating set cover,” *J. ACM*, vol. 45, no. 4, pp. 634–652, 1998.

[28] P. Chalermsook, M. Cygan, G. Kortsarz, B. Laekhanukit, P. Manurangsi, D. Nanongkai, and L. Trevisan, “From gap-ETH to FPT-inapproximability: Clique, Dominating set, and more,” *CoRR*, vol. abs/1708.04218, 2017.

[29] J. Chen, X. Huang, I. A. Kanj, and G. Xia, “Linear FPT reductions and computational lower bounds,” in *STOC*, 2004, pp. 212–221.

[30] C. Lund and M. Yannakakis, “On the hardness of approximating minimization problems,” *J. ACM*, vol. 41, no. 5, pp. 960–981, 1994.

[31] R. H. Chitnis, M. Hajiaghayi, and G. Kortsarz, “Fixed-parameter and approximation algorithms: A new look,” in *IPEC*, 2013, pp. 110–122.

[32] D. Moshkovitz, “The projection games conjecture and the np-hardness of ln n-approximating set-cover,” *Theory of Computing*, vol. 11, pp. 221–235, 2015.

[33] Y. Chen, M. Grohe, and M. Grüber, “On parameterized approximability,” in *IWPEC*, 2006, pp. 109–120.

[34] R. Impagliazzo, R. Paturi, and F. Zane, “Which problems have strongly exponential complexity?” *J. Comput. Syst. Sci.*, vol. 63, no. 4, pp. 512–530, 2001.

[35] M. Charikar, M. Hajiaghayi, and H. J. Karloff, “Improved approximation algorithms for label cover problems,” *Algorithmica*, vol. 61, no. 1, pp. 190–206, 2011.

[36] S. P. Vadhan, “Pseudorandomness,” *Foundations and Trends in Theoretical Computer Science*, vol. 7, no. 1-3, pp. 1–336, 2012.

[37] M. Bellare, O. Goldreich, and M. Sudan, “Free bits, PCPs, and nonapproximability-towards tight results,” *SIAM J. Comput.*, vol. 27, no. 3, pp. 804–915, 1998.

[38] D. Zuckerman, “Linear degree extractors and the inapproximability of max clique and chromatic number,” *Theory of Computing*, vol. 3, no. 1, pp. 103–128, 2007.

[39] R. G. Downey, M. R. Fellows, C. McCartin, and F. A. Rosamond, “Parameterized approximation of dominating set problems,” *Inf. Process. Lett.*, vol. 109, no. 1, pp. 68–70, 2008.

[40] B. Applebaum, “Exponentially-hard gap-csp and local PRG via local hardcore functions,” *ECCC*, vol. 24, p. 63, 2017.