Reducing the Optimum: Fixed Parameter Inapproximability for Clique and Set Cover in Time Super-exponential in Optimum

Mohammad T. Hajiaghayi\textsuperscript{1}, Rohit Khandekar\textsuperscript{2}, and Guy Kortsarz\textsuperscript{3}\textsuperscript{⋆⋆}

\textsuperscript{1} University of Maryland at College Park, USA hajiagha@cs.umd.edu
\textsuperscript{2} KCG holdings Inc., USA rkhandekar@gmail.com
\textsuperscript{3} Computer Science Department, Rutgers University, Camden. guyk@camden.rutgers.edu

Abstract. A minimization problem is called fixed parameter $\rho$-inapproximable, for a function $\rho \geq 1$, if there does not exist an algorithm that given a problem instance $I$ with optimum value $\text{opt}$ and an integer $k$, either finds a feasible solution of value at most $\rho(k) \cdot k$ or finds a certificate that $k < \text{opt}$ in time $t(k) \cdot |I|^{\mathcal{O}(1)}$ for some function $t$. For maximization problem the definition is similar. We motivate the study of inapproximability in terms of the parameter $\text{opt}$, the optimum value of an instance. A problem is called $(r,t)$-\textsc{FPT}-hard in parameter $\text{opt}$ for functions $r$ and $t$, if it admits no $r(\text{opt})$ approximation that runs in time $t(\text{opt}) |I|^{\mathcal{O}(1)}$. To prove hardness, we use gap reductions from 3-Sat and assume the Exponential Time Hypothesis (ETH). If the value of $\text{opt}$ is known for the ‘yes’ instance, inapproximability w.r.t. $\text{opt}$ implies inapproximability w.r.t. input integer $k$ but not vice versa. Hence inapproximability in $\text{opt}$ is stronger. Previous FPT Hardness results \cite{2} have running sub-exponential in $\text{opt}$ which is just a “translation” of inapproximability results to FPT-hardness. In this paper we are only interested in times $t(\text{opt})$ that are super-exponential in $\text{opt}$. Fellows \cite{7} conjectured that \textsc{setcover} and \textsc{clique} are $(r,t)$-\textsc{FPT}-hard for any pair of non-decreasing functions $r,t$ and input parameter $k$. We give the first inapproximability results for these problems with running times super-exponential in $\text{opt}$. Our paper introduces systematic techniques to reduce the value of the optimum. These techniques are robust and work for three quite different problems. In particular one of our results shows that, under ETH, \textsc{clique} is $(r,t)$-\textsc{FPT}-hard for $r(\text{opt}) = 1/(1 - \epsilon)$ with some constant $\epsilon > 0$ and any non-decreasing function $t$. The running time can be also set to $2^{o(n)}$, for an arbitrary $o(n)$ exponent. This improves the main result of Feige and Kilian \cite{6} in two ways. We also show that the Minimum Maximal Independent Set (\textsc{mmis}) problem is $(r,t)$-\textsc{FPT}-hard in $\text{opt}$, for arbitrarily fast growing functions $r,t$ of $\text{opt}$. In terms of $k$ it was proven \cite{4} that that a $(r(k),t(k))$ approximation is $W[2]$-hard, which is a far stronger assumption than ours (in fact, the hardness of \cite{4} can indirectly be proved under ETH by combining \cite{4} with several other papers). This part is technically very complex, and the reduction of $\text{opt}$ is very instructive and we believe will find further applications.

\textsuperscript{⋆} Supported in part by NSF CAREER award 1053605, NSF grant CCF-1161626, ONR YIP award N000141110662, and DARPA/AFOSR grant FA9550-12-1-0423. Email: hajiagha@cs.umd.edu.

\textsuperscript{⋆⋆} Partially supported by NSF award number 1218620.
1 Introduction

1.1 FPT inapproximability with parameter opt: motivation

An FPT-approximation algorithm for a minimization (resp., maximization) problem $P$, is called an $(r,t)$-FPT-approximation algorithm for $P$ with input parameter $k$, if the algorithm takes as input an instance $I$ with value $\text{opt}$ and an integer parameter $k$ and either computes a feasible solution to $I$ with value at most $k \cdot r(k)$ (resp., at least $k/r(k)$) or computes a certificate that $k < \text{opt}$ (resp., $k > \text{opt}$) in time $t(k) \cdot |I|^{O(1)}$. In the latter case, such a certificate can be obtained from the analysis of the algorithm and the fact that it did not produce the desired solution. Here the goal is to design algorithms with as slow growing functions $r$ and $t$ as possible. A problem is called $(r,t)$-FPT-inapproximable (or, $(r,t)$-FPT-hard) if it does not admit any $(r,t)$-FPT-approximation algorithm. Here the goal is to show inapproximability with as fast growing functions $r$ and $t$ as possible. In this paper, we use reductions from $3$-SAT to prove FPT-hardness.

When doing a reduction from $3$-SAT to an optimization problem $P$, a “yes” instance of $P$ is an instance obtained from a satisfiable formula, and a “no” instance of $P$ is an instance obtained from a reduction of non-satisfiable formula. Whenever we use reductions from $3$-SAT, we denote the number of variables by $q$, the number of clauses by $m$ and $N = m + q$. We always use $n$ to describe the size of the problem we reduce to. The following conjecture is one of the leading challenge in the theory of fixed parameter tractability. Mike Fellows, conjectured the following for SETCOVER and CLIQUE.

**Conjecture 1 (FPT-hardness of SETCOVER and CLIQUE (Fellows [7])).**

- SETCOVER is $(r,t)$-FPT-hard for any non-decreasing functions $r$ and $t$.
- CLIQUE is $(r,t)$-FPT-hard for any non-decreasing functions $r(k) = o(k)$ and $t$.

When we do a gap reduction from $3$-SAT, in all cases in this paper, the optimum $\text{opt}$ of a yes instance is known. Thus, by definition, inapproximability in terms of $\text{opt}$ implies inapproximability in terms of $k$ by setting $k = \text{opt}$. On the other hand, inapproximability in terms of $k$ does not imply inapproximability in $\text{opt}$ because it may be that $k < \text{opt}$. Thus our inapproximability in $\text{opt}$ is strictly stronger than in $k$ (as $\text{opt}$ in our paper, always happens to be known).

**Conjecture 2 (FPT-hardness of SETCOVER and CLIQUE with parameter OPT).** Let $\text{opt}$ and $n$ denote the value of the optimum and size of the given instance, respectively.

- SETCOVER admits no $r(\text{opt})$ approximation that runs in time $t(\text{opt}) \cdot n^{O(1)}$ for any non-decreasing functions $r$ and $t$.
- CLIQUE admits no $r(\text{opt})$ approximation that runs in time $t(\text{opt}) \cdot n^{O(1)}$ for any non-decreasing functions $r(k) = o(k)$ and $t$.

If we do not insist on super exponential time, proving the FPT-hardness is a direct translation of the inapproximability hardness.
1.2 Reducing the value of OPT

We make a very important breakthrough proving hardness results for setcover and clique with running times $t(\text{OPT})$ that are super-exponential in $\text{OPT}$. Such results were never proven before. In particular, the inapproximability of [2] is under a sub-exponential running time. We discuss this and the relation to the reducing the value of OPT. Proving hardness result with super-exponential time in OPT or $k$ necessitates reducing the value of OPT. The reason is that $t(\text{OPT}) = 2^{o(\alpha m)}$ must hold with $m$ the number of clauses in the 3-SAT instance we reduce from. As the function $t$ becomes faster and faster growing, OPT needs to get smaller and smaller. Therefore we claim that on of the most important aspects of the art of proving FPT-hardness is creating new instances with smaller and smaller OPT. We develop systematic techniques to reduce the value of OPT for the problems we study. No FPT hardness before us used any techniques to reduce OPT, even though reducing the OPT seems to us to be one of the most important ideas for proving FPT-hardness.

We also study a problem called the Minimum Maximal Independent Set (MMIS) problem. In this problem, given an undirected graph, the goal is to find a minimum-size independent set that is also inclusion-wise maximal.

2 Previous work

The following relation is known among the parameterized complexity classes: FPT $\subseteq$ W[1] $\subseteq$ W[2]. It is widely believed that FPT $\neq$ W[1]. In fact FPT=W[1] implies that ETH fails. The Projection game conjecture (PGC) [10] is as follows.

**Conjecture 3.** There exists a constant $c$ so that for every $\epsilon > 1/n^c$ there is a PCP of almost linear size, namely, size $[I] \cdot 2^{\log^* |I|} \cdot \text{poly}(1/\epsilon)$ for some constant $0 < \alpha < 1$, with error $1/\epsilon$ and alphabet size $\text{poly}(1/\epsilon)$

This conjecture is known to be valid, if alphabet has size $exp(1/\epsilon)$. See [11]. We note that in [10] the sublinear term is not specified. Hence we did the natural thing and took the sublinear term from [11].

**Theorem 1.** [2] Under ETH and PGC, there exist constants $1 > F1, F2 > 0$ such that the setcover problem does not admit an FPT approximation algorithm with ratio $OPT^{F1}$ in time $2^{opt^{F2}} \cdot \text{poly}(n)$.

**Theorem 2.** [2] Unless NP $\subseteq$ SUBEXP, for every $0 < \delta < 1$ there exists a constant $F = F(\delta) > 0$ such that clique admits no FPT approximation within $OPT^{1-\delta}$ in time $2^{opt^{F}} \cdot \text{poly}(n)$.

In both results the time is strictly subexponential in OPT.

**Previous work on MMIS:** In [4] it is proven that for any increasing $r,s$, and $(r(k), s(k))$ approximation is W[2]-Hard. We prove a stronger statement namely $(r(\text{OPT}), t(\text{OPT}))$-FPT-hardness for any increasing $r,s$ (which is stronger than inapproximability in $k$) and this is under the ETH which is a much weaker assumption that FPT $\neq$ W[2]. More precisely, our hardness has the advantage
of being self contained, because it is possible to get \((r(k), s(k))\)-hardness under \textsc{eth}, by taking \([4]\) and combining it with several other papers that together prove that \(\text{FPT} = W[2]\) contradicts the \textsc{eth}.

Most importantly, the reducing of the optimum value in \textsc{mmis} is a very hard technical challenge. We are sure that the ideas used for reducing the optimum for \textsc{mmis} will find future applications.

3 Preliminaries

Impagliazzo et al. \([8]\) formulated the following conjecture which is known as \textsc{eth}.

We assume \textsc{eth} in all hardness results in this paper.

<table>
<thead>
<tr>
<th>Exponential Time Hypothesis (\textsc{eth})</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-sat cannot be solved in (2^{o(q+m)}) (O(1)) time where (q) is the number of variables and (m) is the number of clauses.</td>
</tr>
</tbody>
</table>

Using the Sparsification Lemma of Calabro et al. \([1]\), the following lemma follows.

**Lemma 1.** Assuming \textsc{eth}, 3-sat cannot be solved in \(2^{O(m)}(q + m)O(1)\) time where \(q\) is the number of variables and \(m\) is the number of clauses.

**Important remark:** When we use expressions such as, for example, \(\text{poly}(1/\epsilon)\) in a “yes” and a “no” instance, the terms are always identical, hence a gap holds.

From \([9]\) \([10]\) the following is derived.

**Corollary 1.** There exists a reduction from 3-sat to \textsc{setcover} with \(m\) clauses so that

1. The number of sets is \(m \cdot 2^{\log^\alpha m} \cdot \text{poly}(m) \cdot \text{poly}(1/\epsilon)\) and \(\alpha\) a constant that satisfies \(0 < \alpha < 1\).
2. The number of elements is \(\text{poly}(m)\).
3. The optimum for a “yes” instance is exactly \(m \cdot 2^{\log^\alpha m} \cdot \text{poly}(m) \cdot \text{poly}(1/\epsilon)\).
4. The optimum for a “no” instance is at least \(d \cdot \sqrt{\epsilon} \cdot m \cdot 2^{\log^\alpha m} \cdot \text{poly}(m) \cdot \text{poly}(1/\epsilon)\) for a constant \(d > 0\).

4 Our results

In all our reductions \(\text{OPT}\) is known and so the reduction are stronger than reductions in \(k\).

**Theorem 3.** Under \textsc{eth} and \textsc{pgc}, \textsc{setcover} is \((r, t)\)-\text{FPT-hard} for \(r(\text{OPT}) = (\log \text{OPT})^\gamma\) and \(t(\text{OPT}) = \exp(\exp((\log \text{OPT})^\gamma))\) \(\cdot \text{poly}(n)\) \(= \exp \left( \text{OPT}^{\gamma} \cdot \text{OPT} \right) \cdot \text{poly}(n)\) for some constant \(\gamma > 1\) and \(f = \gamma - 1\).

The time here is much larger than just exponential in \text{OPT}. Further \(\Omega(\log \text{OPT})\) hardness follows trivially from the known hardness for \textsc{setcover} \([5]\). We prove a stronger hardness as \(\gamma > 1\).
Theorem 4. Under ETH and a stronger version of PGC with PCP length $O(m \cdot \text{poly log}(m) \cdot \text{log}(1/\epsilon))$ and gap $\Omega(1/\epsilon)$ for $\epsilon \geq 1/m^c$, for some constant $c$, setcover is $(r,t)$-FPT-hard for $r(\text{OPT}) = \text{OPT}^{d'}$ and $t(\text{OPT}) = \exp(\exp(\text{OPT}^{d''})) \cdot \text{poly}(n)$ for some constants $d', d'' > 0$.

This kind of PCP was conjectured to exist by Moshkovitz in a private communication.

Note that the running times in this result is almost doubly exponential in OPT.

We can also prove an inapproximability with super-exponential time in OPT that only assumes ETH.

Theorem 5. Under ETH alone, setcover cannot be approximated within $c \sqrt{\log \text{OPT}}$ for some constant $c$, in time $\exp\left(\text{OPT}^{(\log \text{OPT})^{f}}\right) \cdot \text{poly}(n)$ for $f$ the same constant from Theorem 3.

Theorem 6. Under ETH, clique is $(r,t)$-FPT-hard for $r(\text{OPT}) = 1/(1 - \epsilon)$ for some constant $\epsilon$, that satisfies $0 < \epsilon < 1$, and any non-decreasing function $t$, however huge. The running time can also be set to $2^{o(n)}$ of our choice of $o(n)$.

It is interesting to compare this result to the paper by Feige et al [6]. In [6] it is shown that if $\text{OPT} \leq \log n$ and clique problem can be solved in time significantly smaller than $n^{\text{OPT}} < n^{\log n}$, then any NPC problem that uses $f(n)$ non-deterministic bits can be solved in time roughly $\exp(\sqrt{f(n)})$. Among other things, this implies that 3-SAT can be solved in time roughly $\exp(\sqrt{n})$ which contradicts the ETH. Hence the the assumption in [6] is weaker and implies our assumption, namely the ETH.

Theorem 6 works for any OPT and $\text{OPT} \leq \log n$ in particular, and thus improves the paper of Feige et al [6] in two ways. First we prove $1/(1 - \epsilon)$-hardness which for such small values of OPT might be significantly harder than ruling out an exact solution. Second, the $r(\text{OPT})$-hardness holds even if we allow time $2^{n^{o(n)}}$ time. It may seem strange that we can get FPT-hardness in such high running time. The "trick" is that the first step we do, is transforming the graph to a new one, of size $2^{n^{o(n)}}$. This time, $2^{o(n)}$ strongly improves the time $n^{\log n}$ of [6].

Theorem 7. Let

$$r(\text{OPT}) = \left(\frac{1}{1 - \epsilon}\right)^{\log^{1/3} \text{OPT}},$$

with $\epsilon$ the constant from Theorem 6. Then clique is $(r,t)$-FPT-hard, for any function $t$, however huge.

As a function of $n$, we later show that the time can be set to $2^{n^{1/Q(n)}}$ for an arbitrarily slowly growing $Q(n)$. Thus Theorem 7 improves [6] in the same two ways that we mentioned for Theorem 6. The inapproximability is now super constant, versus an exact solution, and the running time is still much much higher.

Theorem 8. Under the ETH, mmis is $(r,t)$-FPT-hard in OPT (and thus in $k$ since OPT is known) for any non-decreasing functions $r$ and $t$. 
5 Inapproximability for Set Cover with super-exponential time in OPT

In this section we prove Theorem 3.

Corollary 1 implies the following corollary.

Corollary 2. There exists $0 < \alpha < 1$, so that the following holds. Let $m$ be the number of clauses in the 3-SAT problem we reduce from. Assuming P≠NP and ETH, there exists a reduction from 3-SAT to SETCOVER so that the number of sets in the resulting instance is $\sigma = m \cdot 2^{\log^\alpha m} \cdot \text{poly}(\log(m)) \cdot \text{poly}(1/\epsilon)$. Furthermore, more, value of the optimum in “yes” instance is exactly $\kappa = m \cdot 2^{\log^\alpha m} \cdot \text{poly}(\log(m))\text{pol}(1/\epsilon)$ and that in the “no” instance is at least $c \cdot \log m \cdot \kappa$.

We use $c = \epsilon^2/\log^2 m$ here (the least value for which the proof works).

We now describe a way to change the SETCOVER. The idea is to make the optimum much smaller. Starting with the SETCOVER instance $S = (U, \mathcal{S})$ in the above corollary, where $U$ is the set of elements and $\mathcal{S} \subseteq 2^U$ is the collection of sets, we construct a new instance $S' = (U, \mathcal{S}')$ on the same elements as follows. We introduce a set $s \in \mathcal{S}'$ as $s = \cup_{p=1}^p s_i$ for each subcollection $\{s_1, s_2, \ldots, s_p\} \subseteq \mathcal{S}$ of size $p$ where $1 \leq p \leq \lfloor m/\log m \rfloor$.

Claim. The number of sets in the new instance $S' = (U, \mathcal{S}')$ is $2^{o(m)}$. The new instance can be constructed in time $2^{o(m)}$.

Proof. Recall that the number of sets in the original instance is $\sigma = m \cdot 2^{\log^\alpha m} \cdot \text{poly}(\log(m))$ because of the choice of $\epsilon$. Thus since $p \leq \sigma/2$, the number of sets in the new instance is

$$\sum_{p=1}^{\lfloor m/\log m \rfloor} \binom{\sigma}{p} \leq \lfloor m/\log m \rfloor \cdot \left( m \cdot 2^{\log^\alpha m} \cdot \text{poly}(\log(m)) \right)^{m/\log m} = \Theta(m^{\log^\alpha m} \cdot \text{poly}(\log(m))) = 2^{o(m)},$$

We use the inequality $\binom{\sigma}{p} \leq (\sigma/e)^p$ to upper-bound this by

$$\lfloor m/\log m \rfloor \cdot \left( \epsilon \cdot 2^{\log^\alpha m} \cdot \text{poly}(\log(m)) \cdot 2 \log m \right)^{m/\log m} = 2^{o(m)},$$

as claimed, where we have $2 \log m$ in the first expression (instead of just $\log m$) because of the floor function and the last equality holds since $0 < \alpha < 1$. It is easy to see that the new instance can be created in $2^{o(m)}$ time.

Proof of Theorem 3.

Proof. Clearly, any optimum will use as few sets as possible $m/\log m$ and so the gap between a “Yes” instance and a “No” instance changed. Namely, $\text{OPT}_1$ and that of the new instance $\text{OPT}_2$ are related as $\text{OPT}_2 \leq \lfloor \text{OPT}_1/\lfloor m/\log m \rfloor \rfloor$. Therefore the gap between the new optimum of a “yes” instance and a “no” instance continues to be $c' \log m$ for some constant $c' > 0$ and the new optimum of the “yes” instance is at most $\kappa_T = \lfloor \kappa/\lfloor m/\log m \rfloor \rfloor = O(2^{\log^\alpha m} \cdot \text{poly}(\log(m)))$, and $\kappa_N$ is $c' \cdot \log m$ larger than that.

Now define two functions $r(k) = (\log k)\gamma$ and $t(k) = \exp(\exp((\log k)\gamma))$ for any $1 < \gamma < 1/\alpha$, as given in Theorem 3. Note that $r(k) = O((\log^\alpha m)^\gamma) = o((\log^\alpha m)^{1/\alpha}) = o(\log m)$ and $t(k) = 2^{o(m)}$. Thus SETCOVER is $(r, t)$-FPT-hard for these functions, proving Theorem 3.
5.1 Proof of Theorem 4

For proving this theorem we assume:

Conjecture 4. There exists a constant $c > 0$ and a PCP of size $m \cdot \text{poly log}(m) \cdot \text{poly}(1/\epsilon)$, for any $\epsilon$ so that $\epsilon \geq 1/m^c$.

Is the conjecture reliable?

This result was conjectured to hold by Moshkovitz in a private communication. Note that there exists already a PCP of size even smaller than the above. In fact in [3] a PCP is presented whose size is $m \cdot \text{poly log}(m)$. The down size is that the inapproximability of this PCP [3] is 2. Improving the inapproximability to polylogarithmic does not seem far fetched.

We now use the above conjecture and show a much stronger FPT inapproximability for SETCOVER. By Corollary 1, and the above conjecture we get the following corollary, using $\epsilon = \epsilon^2/\log^2 m$:

Corollary 3. There exists a constants $c,c_1 > 0$ and a reduction from 3-SAT to SETCOVER so that:

1. The number of sets is $\sigma = m \cdot \text{poly log}(m) \cdot (1/\epsilon)^{c_1+c_2} = m \cdot \text{poly log}(m)$ for some constants $c_1,c_2$.
2. The number of elements is $\text{poly}(m)$.
3. The value of the optimum in “yes” instance is exactly $\kappa_V = m \cdot \text{poly log}(m)$ and in the “no” instance is at least $c \cdot \log m \cdot \kappa$ with $c$ some constant $c > 0$.

Proof of Theorem 4: Make every collection of sets of size $m/(d \cdot \log \log m)$ one big ‘collection set’, with $d$ a large enough constant. Here we omit the floor and the ceiling as, in the previous proof, we saw that they hardly make a difference, and the correction needed is minimal.

The number of sets in the instance is:

$$\left( \frac{m \cdot \text{poly log}(m)}{d \log \log m} \right)$$

and is $2^{o(m)}$ if $d$ is large enough. This is implied by the inequality $\binom{n}{k} \leq (ne/k)^k$.

The reason for the major improvement is that the term $2^{\log^x m}$ is gone.

After this change, the size of the optimum for a “yes” instance becomes $\text{poly log}(m)$. Recall that the gap is $c \log m$. Therefore, the gap can be stated as $\text{OPT}^d$ for some $d' < 1$.

Let $d''$ be any constant $d'' < d'$. As for the running time, we use $\text{OPT}^d = c \log m$, we get $2^{\text{OPT}^{d''}} = o(m)$ and $\exp(2^{\text{OPT}^{d''}}) = 2^{o(m)}$. This ends the proof of Theorem 4.

5.2 An inapproximability under the Exponential Time Hypothesis only

For lack of space the proof of this theorem is given in Appendix C.
6 A constant lower bound for Clique in arbitrarily large time in opt

We use the basic PCP theorem:

**Theorem 9 (The standard pcp theorem).** There exists a reduction from any NP-complete language $L$ to 3-SAT so that a “yes” instance is mapped to a 3-SAT instance such that all clauses can be simultaneously satisfied, while a “no”-instance is mapped to an instance such that at most $1 - \epsilon$ fraction of the clauses can be simultaneously satisfied. Here $\epsilon > 0$ is a constant.

The following standard claim, follows from the PCP theorem.

**Claim.** There exist a positive $\epsilon > 0$ and a gap preserving reduction from a 3-SAT instance, with $q$ variables and $m$ clauses, to an instance of clique with $n = 7m$ vertices, so that for a “yes” instance, the corresponding clique instance, has clique of size $m$, and for a “no” instance, the instance has a maximum clique of size at most $(1 - \epsilon)m$.

We do the following transformation that is a modification of what we did for setcover. The number of vertices in the clique instance is $7m$. Let $f(m)$ be any slowly non-decreasing function of $m$ such that $f(m) = \omega(1)$. First, note that we may assume that $m$ is divisible by $f(m)$ without loss of generality. Indeed, we need to add fake clauses to the 3-SAT instance of the type $(x \lor \bar{x} \lor z_1), (x \lor \bar{x} \lor z_2), \ldots$ so that the number of clauses added is at most $f(m)$ and we make $m$ divisible by $f(m)$. Since $f(m)$ is very small compared to $m$, this makes no difference. We create a new clique instance by introducing a vertex for each subset of size $m/f(m)$ vertices in the old clique instance. Such a vertex is called a ‘supervertex’. Two supervertices $A, B$, are connected by an edge, if $A \cup B$ is a clique, and $A \cap B = \emptyset$. The last condition, namely, the fact that two sets that are connected must be disjoint is not needed in the setcover reduction, but it is crucial here.

**Claim.** The new instance of the clique problem has size $2^{q(m)}$.

**Proof.** Using $\binom{n}{2} \leq (ne/k)^k$, we get that the number of supervertices is at most $(7e \cdot f(m))^{7m/f(m)} = \exp(\log(7e \cdot f(m)) \cdot 7m/f(m)) = 2^{q(m)}$, since $f(m) = \omega(1)$. The number of edges in the new clique instance, being at most the square of the number of vertices, is also $2^{q(m)}$.

**Claim.** The maximum clique size in any new instance is exactly $f(m)$. The gap between the clique sizes of the new “yes” and “no” instances is $1/(1 - \epsilon)$, which implies $1/(1 - \epsilon)$-hardness.

**Proof.** Since the maximum clique size in the old instance is $m$, we get that the maximum clique size in the new instance is $f(m)$. Indeed, we can take the optimum clique and divide it into $m/f(m)$ disjoint sets. By the chosen size these sets are supervertices and their union is the old optimum clique. This shows that the new size of the clique is at least $f(m)$. Since two distinct collection vertices $A$ and $B$ are adjacent in the new instance, only if $A \cup B$ is a clique, and $A, B$ are disjoint, it follows that the largest clique size of the new “yes” instance is...
exactly \( f(m) \) because taking more than \( f(m) \) disjoint sets gives a clique of size larger than \( m \), contradicting the fact that \( m \) is the maximum size of the clique. Thus for a yes instance \( f(m) \) is the new size of the maximum clique.

The maximum clique in the new “no” instance, on the other hand, is at most \((1 - \epsilon) m / (m / f(m)) = f(m)(1 - \epsilon)\), otherwise there would exist a clique in the old instance of size larger than \((1 - \epsilon) m\). The proof is thus complete.

**Claim.** The time can be set to be \( t(\text{opt}) \cdot n^{O(1)} \) for any non-decreasing function \( t \).

**Proof.** Since \( f(m) \) can be as small as we wish, we can make the time \( t(f(m)) \) as small as we want. Let \( h(\text{opt}) = 2^{\text{opt}} \). Selecting \( f(m) = t^{-1}(h(m)) \) gives \( h(m) = 2^{\text{opt}} \) time. Since \( m, n \) are linearly related here the time can be set to \( 2^{\text{opt}} \) for any \( t \).

### 6.1 A super constant inapproximability

In this section, we use graph products to prove a super constant inapproximability for \textsc{clique} in time arbitrarily large in \( \text{opt} \). Due to space limitation, the proof is given Section B in the appendix.

### 7 FPT Hardness for Minimum Maximal Independent Set

In this section we prove Theorem 8. We start with 3-Sat instance \( I \) with \( m \) clauses and \( q \) variables. We assume that a “yes” instance admits a satisfying assignment and in the case of a “no” instance, any assignment will leave at least one clause unsatisfied. We now describe how to build the new graph \( G(I) = (V(I), E(I)) \).

**The building blocks:**

1. For every variable \( x \) in \( C \), we define two vertices \( u_x \) and \( \bar{u}_x \). The choice of a vertex \( u_x \) represents an assignment True to \( x \) and the choice of \( \bar{u}_x \) represents a False assignment to \( x \).
2. For every clause we add a set \( W(C) \) of \( q \) copies of the clause. Namely, \( W(C) = \{w^C_x, \ldots, w^C_q\} \).

Intuitively, we want to create a \textsc{setcover}-like instance in which variables are sets and clauses are elements and a variable \( u_x \) covers \( C \) if \( x \in C \) and \( \bar{u}_x \) covers \( C \) if \( \bar{x} \in C \).

**Supervertices:** Similar to our construction for \textsc{setcover} and \textsc{clique}, we define a new graph \( H(I) \) with supervertices that are collections of vertices of the type \( u_x, \bar{u}_x \). Let \( f(q) \) be any slowly increasing function of \( q \) such that \( f(q) = \omega(1) \) and assume, by adding dummy clauses if needed, that \( f(q) \) divides \( q \). The supervertices of \( V(I) \), denoted by \( v_S \), correspond to subsets \( S \subseteq \{u_x \mid x \in C\} \cup \{u_x \mid x \in C\} \) satisfying the following two conditions:

1. \( |S| = q / f(q) \),
2. $S$ does not contain both $u_z, \bar{u}_z$ for any variable $z$ (i.e., a set $S$ does not contain a “contradiction” in the truth value assignment).

**Edges between two supervertices:** Introduce an edge between $v_{S_1}$ and $v_{S_2}$ if and only if there exists some variable $x$ so that either $u_x$ or $\bar{u}_x$ belongs to $S_1$ and either $u_x$ or $\bar{u}_x$ belongs to $S_2$ Note that the above gives four cases in which $v_{S_1}, v_{S_2}$ are connected.

**Edges between supervertices and $W(C)$ vertices:** Introduce edges as follows:

1. If a variable $x \in C$, any supervertex that contains the vertex $u_x$ is connected to all vertices of $W(C)$.
2. If a variable $\bar{x} \in C$, any supervertex that contains $\bar{u}_x$ is connected to all vertices of $W(C)$.

**Example:** Say for example $C = (x \lor \bar{z} \lor w)$. Then any supervertex that contains $u_x$ is connected to all the copies of $W(C)$. Also, every supervertex that contains $\bar{u}_x$ or $u_w$ is connected to all the copies of $W(C)$.

What complicates this are two factors:

1. The supervertices chosen have to be an independent set $I$ in $G(I)$
2. All Supervertices not chosen have to have a neighbor in $I$

**Claim.** Total number of vertices in $H(I)$ is $2^\omega(q) + qm$. The instance $G(I)$ can be constructed in time $2^\omega(q)$.

**Proof.** The total number of vertices in $H(I)$ of type $v_S$ for $S \subset A$ is at most \((\frac{q}{f(q)}) < (qe/(q/f(q))^{q/f(q)} < 2^\omega(q))\). Here we again use the inequality \(\binom{n}{k} \leq (qe/k)^k\). The number of vertices of type $W(C)$ for a clause $C$ is $qm$.

**Building an MMIS of size $f(q)$ for a “yes” instance:**

1. Start with the set $X = \{u_x \mid x$ is a literal$\}$. This set contains for every variable its vertex copy that corresponds to a True assignment.
2. Decompose $X$ to $f(q)$ pairwise disjoint sets each containing $q/f(q)$ vertices. Let these sets be $S_1, S_2, \ldots, S_{q/f(q)}$. We want to derive sets so that $v_{S_i}$ is a feasible MMIS, which is of course not the case so far (because not all $W(C)$ are covered).
3. We now modify sets $S_i$ to obtain sets $T_i$ as follows. Fix a satisfying assignment $\tau$ to the variables. We start by setting $T_i = S_i$ for all $i$. If $\tau(x)$ is False, then for the unique $i$ so that $u_x \in T_i$, remove $u_x$ from $T_i$ and add $\bar{u}_x$ to $T_i$.

This is done for all variables. The final $T_i$ sets are called the assignment sets.

Our solution will be $I = \{v_{T_i} \mid T_i$ is an assignment set$\}$.

**Claim.** The set $\{v_{T_i}\}$ is independent in $H(I)$.

**Proof.** For the vertices $v_{T_i}, v_{T_j}$ with $i \neq j$ to be connected it must be that some $x$ so that either $u_x$ or $\bar{u}_x$ belongs to $T_i$ and either $u_x$ or $\bar{u}_x$ belongs to $T_j$. Clearly, this implies that $u_x \in S_i \cap S_j$. This is a contradiction to the fact that the sets $\{S_p\}$ are pairwise disjoint.
Claim. The \( f(q) \) vertices \( \{v_T\} \) defined above form a dominating set in \( H(I) \).

Proof. We first show each vertex in \( W(C) \) is adjacent to some vertex \( v_T \). Note that \( \tau \) satisfies all clauses \( C \). One possibility is that \( \tau(x) \) is True and \( x \in C \). Thus the unique assignment set \( T_i \) that contains \( u_x \) is connected to all the copies \( W(C) \) of \( C \). Alternatively, if \( \tau(x) \) is False and \( \bar{x} \in C \), the unique \( T_i \) that contains \( \bar{u}_x \) is connected to all copies of \( W(C) \).

We now show that \( I \) dominates every supervertex not in \( I \). Let \( v_S \) be a vertex of \( H(I) \) that does not belong to \( I \). Pick an arbitrary variable \( x \) so that either \( u_x \in S \), or \( \bar{u}_x \in S \). By construction there is some assignment set \( T_i \in I \) that contains \( u_x \) or \( \bar{u}_x \). In all the four cases above, by definition, there is an edge between \( v_T \) and \( v_S \).

Thus we just proved the following corollary.

Corollary 4. The “yes” instance admits a solution of size \( f(q) \).

Claim. For a no instance the minimum MMIS is of size larger than \( q \).

Proof. Let \( S \) be the optimum MMIS of the “no” instance. Note that all super vertices chosen by the optimum have to be consistent. Namely, we can not have \( u_x \) belonging to one set \( T_i \) in \( S \) and \( \bar{u}_x \) to some \( T_j \in S \) because this will imply an edge between \( v_T \) and \( v_T \) and a contradiction. In particular, this implies that vertices \( \{v_T\} \) represent a (maybe partial) truth assignment to the variables. Since we are dealing with a no instance, there must be a clause \( C \) that is not satisfied by this partial assignment. This means that none of the vertices that correspond to literals that satisfy \( C \) are in any set of \( S \). For example if \( C = (x \lor \bar{z} \lor w) \) then there may be one set related to \( x \) but it contains \( \bar{u}_x \), because the assignment does not satisfy \( C \). There may be one set related to \( z \), but it contains \( u_z \), and there may be a set for \( w \), but it contains \( \bar{u}_w \). This means that the \( q \) copies \( W(C) \) must be present in \( S \), since it is a maximal independent set. Thus the size of \( S \) is at least \( q \).

Theorem 10. Assuming the ETH, MMIS problem is \((r,t)\)-hardness, for any \( r,t \).

Proof. Since the new optimum for a yes instance is \( f(q) \) where \( f \) is an arbitrarily slow growing function. For any given functions \( r \) and \( t \), we can make sure that \( r(f(q)) < q/f(q) \) and \( t(f(q)) = 2^{\omega(q)} \). Note that by Claims 7 and 7, the gap between “yes” and “no” instances is larger than \( q/f(q) \). If there existed an \((r,t)\)-FPT-approximation for MMIS, we could distinguish between a “yes” and a “no” instance of \( 3\text{-sat} \) in time \( 2^{\omega(q)} \), contradicting the ETH.

8 Summary

We were able to give the first super exponential FPT-hardness for the two central problems, CLIQUE and SETCOVER. It is our hope that this paper will inspire many more FPT hardness result, in time super exponential in \( \text{OPT} \). However, it may be that in order prove Fellows conjecture, a fixed parameter version of the PCP is needed.
A Why some reductions in time sub exponential in opt are weak results

Consider a polynomial time gap reduction from 3-sat to any other problem \( P \). Let \( |I| = q + n \) be the size of the 3-sat instance and thus the size of the instance of \( P \) is clearly bounded by \( m^c \), for some constant \( c \).

Many times the gap with respect to \( \text{opt} \) and the gap with respect to \( n \), are about the same, as \( \text{opt} \) is close to \( n \). For example, for all standard reduction from 3sat to clique and setcover, \( n \) and \( \text{opt} \) are very close. However, there are even cases in which \( \text{opt} \) is much smaller than \( n \). Thus a gap in \( n \) implies a larger gap in \( \text{opt} \).

In any case, say that we have \( \rho(\text{opt}) \) gap that resulted from a reduction from 3-sat. Thus the ETH implies that we cannot find an approximation better than \( \rho(\text{opt}) \) in time \( 2^{o(m)} \).

Thus, all we need to do is to translate \( 2^{o(m)} \) to a function of \( \text{opt} \). In almost all cases \( \text{opt} \leq n \) (albeit if \( \text{opt} \) is polynomial in \( n \) the next claim still holds). In that case, \( \text{opt} \leq n = m^c \). For any constant \( c' > c \), \( 2^{\text{opt}^{1/c'}} = 2^{o(m)} \). Thus we automatically get a \( \rho(\text{opt}) \)-FPT-hardness in time \( 2^{o(n^{1/c'})} \). Clearly, the meaning of such a reduction is limited and it is a translation of the hardness result to FPT-hardness language. The detail that allows us to give such a translation is that \( t(\text{opt}) \) is subexponential in \( \text{opt} \).
B Proof of Theorem 7

Let the graph that we built in previous subsection (whose optimum for a “yes” instance was \( f(m) \)) be denoted \( H(V,E) \). Recall that its size is:

\[ 2^{2 \log(7 \cdot e \cdot f(m)) \cdot 7m/f(m)}. \]

We now recall the power of a graph \( H(V,E) \). We assume the graph is simple, namely has no loops or parallel edges.

**Definition 1.** The graph \( H^k \) has all the tuples \((v_1,v_2,\ldots,v_k)\) so that any \( v_i \) is a vertex of \( V \). The edges are defined as follows. A tuple \((u_1,u_2,\ldots,u_k)\) is joined to \((v_1,v_2,\ldots,v_k)\) if and only iff for \( i = 1 \) to \( k \), either \((u_i,v_i) \in E\) or \( u_i = v_i \).

Note that two different vertices in \( H^k \) have to differ in at least one tuple value.

The following theorem is folklore. Let \( \omega(H) \) be the size of the clique in \( G \).

**Theorem 11.** \( \omega(H^k) = \omega(H)^k \).

To get a super constant gap we take the graph \( H(V,E) \) of previous section and raise it to the power \( \sqrt{f(m)} \). The choice of \( \sqrt{f(m)} \) is rather arbitrary. Recall that for a “yes” instance \( \omega(G) = m \), with \( m \) the number of clauses in the 3-SAT instance and for a “no” instance \( \omega(G) \leq (1 - \epsilon)m \). Hence \( m = \text{OPT} \) for a “yes” instance. Taking this graph to the \( \sqrt{f(\text{OPT})} \) value we get that:

**Corollary 5.** For \( H(V,E)^{\sqrt{f(\text{OPT})}} \), the value of the clique for a “yes” instance is \( f(\text{OPT})^{\sqrt{f(\text{OPT})}} \) and for a “no” instance at most \((1 - \epsilon)\sqrt{f(\text{OPT}) \cdot \text{OPT}^{\sqrt{f(\text{OPT})}}} \).

Note that the new size of the graph is:

\[ 2^{2 \sqrt{f(m)} \log(7 \cdot e \cdot f(m)) \cdot 7m/f(m)} = 2^{o(m)} \]

In addition, the gap is now \( r(\text{OPT}) = (1/(1 - \epsilon)) \sqrt{f(m)} \). We now describe the gap as a function of the new optimum. The optimum for a “yes” instance is \( \text{OPT}' = f(m) \sqrt{f(m)} \). Thus \( (\log \text{OPT}')^{1/3} = \sqrt{f(m)} \). Thus the gap in terms of \( \text{OPT}' \) is:

\[ r(m) = (1/(1 - \epsilon))^{\log^{1/3} \text{OPT}'} \]

**Claim.** Let \( t \) be any non decreasing function and \( r(m) = (1/(1 - \epsilon))^{\log^{1/3} \text{OPT}'} \). Then, Clique is \((r,t)\)-FPT-hard.

**Proof.** The arguments for \( t(\text{OPT}) = 2^{o(m)} \) follows exactly as in Claim 6. Because the new optimum for a “yes” instance \( f(m) \sqrt{f(m)} \), can be made arbitrarily small as well.

Also, as the new \( n \) is \( n' = n \sqrt{f(m)} \) we get \( n = n'^{1/\sqrt{f(m)}} \). As \( f(m) \) can be chosen arbitrarily small, and \( n = 7m \), the time as a function of \( n \) is \( n^{1/Q(n)} \) for any slowly increasing function \( Q(n) \).
C  Hardness of setcover based only on the ETH

For the (maybe unlikely) case that PGC will be proved wrong, we now prove a somewhat weaker inapproximability for SETCOVER assuming ETH only. This result will remain valid even if PGC is disproved.

The following is proved in [11].

**Theorem 12.** There exists a constant $c$ and a PCP of size $m \cdot 2^{\log^* m} \cdot \text{poly}(m)\text{poly}(1/\epsilon)$, such that the size of the alphabet is at most $\exp(1/\epsilon)$ and the gap that can be chosen to be $1/\epsilon$ for any $\epsilon > 1/m^c$.

The difficulty now is that choosing too large $\epsilon$ increases the number of sets very much. Indeed, the number of sets equals the number of vertices in MIN-REP and this number is now

$$m \cdot 2^{\log^* m} \cdot \text{poly}(m)\text{poly}(1/\epsilon) \exp(1/\epsilon).$$

We choose $\epsilon = \ln 2 \cdot \log^* m$. Then using a reduction from 3-SAT to SETCOVER described in Corollary 1 we get:

**Corollary 6.** There exists a constant $d > 0$, and a constant $0 < \alpha < 1$ and a reduction from 3-SAT to SETCOVER so that:

1. The number of sets is $m \cdot 2^{\log^* m} \cdot \text{poly}(m)$.
2. The number of elements is $\text{poly}(m)$.
3. The gap is $d \cdot \sqrt{\log^* m}$.
4. The optimum of a “yes” instance does not change, namely, is $\text{OPT} = 2^{\log^* m} \cdot \text{poly}(m)$.

The proofs here are simple computations using the new value of $\epsilon$ plugged in Corollary 1. The optimum OPT does not change because it does not depend on the alphabet. The reason is, that any optimal solution still takes one vertex from any supervertex hence the optimum for a “yes” instance is still the number of super vertices.

**The inapproximability in terms of opt:** The gap is $d \sqrt{\log^* m}$ for some constant $d$. OPT = $2^{\log^* m} \cdot \text{poly}(m)$. Thus for some constant $c$, the problem is $c \cdot \sqrt{\log \text{OPT}}$-hard.

**The time in terms of opt:** Since OPT did not change we derive exactly the same time as in Theorem 3, namely, $\exp \left( \text{OPT}(\log^* \text{OPT}) \right)$ for the same constant $f > 0$, that appears in Theorem 3.

This proves Theorem 5.