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

Mohammad T. Hajiaghayi1 *, Rohit Khandekar2, and Guy Kortsarz3**

1 University of Maryland at College Park, USA hajiagha@cs.umd.edu
2 KCG holdings Inc., USA rkhandekar@gmail.com
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 \( \tilde{t}(k) \cdot |I|^{O(1)} \) 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. To prove hardness in \( \text{OPT} \), we use gap reductions from 3-SAT and assume the Exponential Time Hypothesis (ETH). If the value of \( \text{OPT}_y \) (the optimum for a yes instance) is known, inapproximability w.r.t. \( \text{OPT} \) implies inapproximability w.r.t. input integer \( k \) but not vise versa. Hence inapproximability in \( \text{OPT} \) is stronger. Previous FPT hardness results [2] have running times sub-exponential in \( \text{OPT} \). Fellows [6] conjectured that SETCOVER and CLIQUE are \( (r, t) \)-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, CLIQUE is \( (r, t) \)-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 [5] in two ways. We prove that the Minimum Maximal Independent Set (MMIS) problem is \( (r, t) \)-FPT-hard in \( \text{OPT} \), for arbitrarily fast growing functions \( r, t \). In terms of \( k \) an elementary reduction [3] shows that there is no \( (r(k), t(k)) \) approximation for any \( r(k), t(k) \) under the assumption \( W[2] \neq \text{FPT} \). The assumption that \( W[2] \neq \text{FPT} \) removes the need to reduce the value of \( k \). Our reduction is significantly harder and technically complex because we need to drastically reduce \( \text{OPT} \) since we assume ETH. The \((f(k), t(k)) \) hardness can be shown under ETH, by combining [3] with several papers showed that the ETH implies that \( W[2] \neq \text{FPT} \). In these papers, indeed, the value of \( k \) is drastically reduced. We hope that our technique to reduce \( \text{OPT} \) for MMIS will find future applications.

* 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.
** 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.

In this paper the goal is to show inapproximability with as fast growing functions $r$ and $t$ as possible, however, with the restriction that $t$ is superexponential function. We reduce from 3-SAT and use the ETH to give the FPT-hardness.

**Definition 1.** Say that we reduce from 3-SAT to a problem $P$. Then a yes instance of $P$ is the one derived from a satisfiable 3-SAT formula and a no instance is the one derived form a unsatisfiable 3-SAT formula.

If we reduce from another NPC problem, we may not be able to use the ETH, hence we always reduce for 3SAT.

Let $P$ be a minimization problem. (The definition for a maximization problem is similar).

**Definition 2.** An $(r(\text{opt}), t(\text{opt}))$ hardness for a problem $P$ is defined as follows. There is a gap reduction from an instance of 3-SAT to $P$ so that:

1. If $\text{OPT}_y$ is the optimum for a yes instance and $\text{OPT}_n$ is the optimum for no instance then: $\text{OPT}_y / \text{OPT}_n < r(\text{OPT}_y)$
2. $t(\text{OPT}_y) = 2^{o(m)}$
3. The value of $\text{OPT}_y$ is known

Thus an $r(\text{OPT}_y)$ approximation in time $t(\text{OPT}_y)$ will imply that we can tell between a yes and a no instance of 3-SAT in time $2^{o(m)}$ contradicting the ETH. The fact that the value of $\text{OPT}_y$ is known, is required for proving that inapproximability in $\text{opt}$ implies an inapproximability in $k$.

**Remark:** While we cant know the value of $\text{opt}$ when designing an algorithm, the value of $\text{OPT}_y$ is often known in gap reduction. For example in SETCOVER, after a reduction from 3-SAT of SETCOVER, $\text{OPT}_y$ equals the number of queries the verifier may ask the first prover plus the number of queries the verifier may ask the second prover. And as the number of queries is known in advance, $\text{OPT}_y$ is known. On the other hand, to the best of our knowledge, there does not exist a reduction from 3-SAT to CLIQUE with gap $n^{1-\epsilon}$ (or larger), in which the value of the optimum CLIQUE in a yes instance is known. All we know is that the value is within some range. Thus in [2], the reduction for SETCOVER were in terms of $\text{OPT}$, however, the reductions for CLIQUE were in terms of $k$, as for this problem $\text{OPT}_y$ is not known.
Proposition 1. Inapproximability in OPT implies inapproximability in $k$ but not vice-versa, in general.

Proof. We can set $k = \text{OPT}_y$. Then the algorithm will return a solution of value $r(\text{OPT}_y) \cdot \text{OPT}_y = r(k) \cdot k$ (as it cannot return $k < \text{OPT}$) and the hardness in $k$ follows. On the other hand, say that we have a hardness result with $k$ and the algorithm returns $k < \text{OPT}$. Clearly, this gives no inapproximability in $\text{OPT}_y$.

If a solution of size $r(k) \cdot k$ is returned, it is not clear if $k \leq \text{OPT}_y$ and thus it is unclear if this gives a bound in term of $\text{OPT}_y$.

2 Comparing FPT-hardness in $k$ and in OPT

It may seem that the distinction between hardness in $k$ and OPT is not significant and in some cases it is not. However, in some cases a reduction in OPT is much more significant than a reduction in $k$. To explain that we consider the following problem. The Minimum size independent dominating (MMSI) problem is to find a minimum size independent set that is also a dominating set. In [3], it is proven that if $W[2] \neq \text{FPT}$, for any increasing $r, s$, the problem is ($r(k), s(k)$)-FPT hard. This reduction is very simple. On the other hand, we give an ($s(\text{OPT}), t(\text{OPT})$)-FPT inapproximability for any increasing $r, t$ under the ETH and this reduction is very complex, and technically challenging. In fact, this reduction is the main technical contribution of our paper. The reason that the reduction of [3] is much simpler is because of the assumption $W[2] \neq \text{FPT}$. Suppose that [3] will be required to prove their theorem under the ETH. To show that the problem can not be approximated within $r(k)$ in time $t(k) \cdot n^{O(1)}$ for any function $t$, it is required to replace $k$ by an arbitrarily slowly increasing $f(k)$, because we need to get $t(f(k)) = 2^{o(m)}$ for every $t$ however huge. Thus the reduction in [3] in fact assumes in advance that no time time $t(k)$ will be enough for approximating the problem. This is the meaning of the assumption $W[2] \neq \text{FPT}$. In contrast we prove that no running time $t(\text{OPT}) \cdot n^{O(1)}$ will be enough and not assume that it will not be enough as in [3]. To do so, we have to drastically reduce the value of $\text{OPT}_y$.

It is more appropriate to compare [3] with our result, if we add to [3] the proof that ETH implies that $W[2] \neq \text{FPT}$. Then the [3] result becomes quite more complex. The proof that ETH implies that $W[2] \neq \text{FPT}$ combines several lemmas that appeared in several papers (see [9] for details). First, it is proved that under the ETH there is no exact solution with time $f(k)^{o(k)}$ for CLIQUE. Then a related problem called the Multicolored Clique is defined. Using the previous theorem it is proved that under the ETH conjecture the Multicolored Clique problem admits no exact $f(k)n^{o(k)}$ time solution. Finally, using the theorem on Multicolored Clique, it is shown that the Dominating Set (and thus SETCOVER) admits no exact $f(k) \cdot n^{o(k)}$ time solution, which clearly implies that $W[2] \neq \text{FPT}$. As we said, proving the hardness in terms of $k$ requires making $k$ very small, and indeed, $k$ is reduced to $f(k)$ for an arbitrarily slowly growing function $f$, in one of these papers that prove that the ETH implies that $W[2] \neq \text{FPT}$ (see [9]).

And the second, and perhaps more important point, is that decreasing $\text{OPT}_y$ to a very small new optimum $\text{OPT}_y'$ requires defining a new instance and proving
that the instance has optimum value $\text{OPT}_y$. Reducing $k$ to $f(k)$ can be done more easily and at time, arbitrarily, since $k$ does not need to be directly related to some input. In summary our reduction for MMIS should not be dismissed as a non surprising corollary of [3] because this is completely false. The fact that we start with the ETH, creates a highly more complex task of reducing $\text{OPT}_y$, and in this case a lower bound in $\text{OPT}$ is much more meaningful than a lower bound in $k$. In fact we suggest that inapproximability should be done with $\text{OPT}$ whenever possible. It is our hope, that in the future people will indeed adopt the inapproximability in $\text{OPT}$. We consider this suggestion as one of the contributions of our paper. Moreover, our techniques for reducing $\text{OPT}_y$ for the MMIS problem, will find more applications in the future because the difficulties we faced are likely to appear in other problems.

Relation of our paper to the paper [2]: The current paper has no intersection with the [2]. In [2] we wanted as large inapproximability in $r$ as possible. Given a reduction with gap $q(n)$ find some $r$ (that is as large as possible) so that $r(\text{OPT}) \leq q(n)$. Since we reduce from 3-SAT, in order to get a contradiction to the ETH, we need to get time $2^{o(m)}$ with $m$ the number of clauses in the 3-SAT instance we start with. If $\text{OPT} \leq \text{poly}(n)$ (which is true in most cases) then for some $c \ \text{OPT} \leq m^c$. Then for any constant $c' > c$, $2^{\text{opt}^{1/c'}} = 2^{o(m)}$. Thus we automatically get a $r(\text{OPT})$-FPT-hardness in time $t(\text{OPT}) = 2^{o(\text{OPT}^{1/c'})}$. This is basically translation of the hardness result to FPT-hardness (albeit it can become a useful and standard technique). In particular, [2] does not need to use almost linear PCP or the PGC at all. In this paper almost linear PCP and the PGC are essential. Also, [2] did not discuss the issue of reducing $\text{OPT}_y$ at all.

Notation: 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. We describe the conjecture in terms of $\text{OPT}$.

Conjecture 1. [FPT-hardness of SETCOVER and CLIQUE]

- SETCOVER is $(r(\text{OPT}), t(\text{OPT}))$-FPT-hard for any non-decreasing functions $r$ and $t$.
- CLIQUE is $(r(\text{OPT}), t(\text{OPT}))$-FPT-hard for any non-decreasing functions $r(\text{OPT})$ and $t$.

Remark: The ratio for CLIQUE must be by definition $o(\text{OPT}_y)$ for otherwise we can return a single vertex and get OPT approximation. In all our reduction OPT$_y$ is known and so a hardness in OPT implies a hardness in $k$.

3 Previous work

The following relation is known among the parameterized complexity classes: $\text{FPT} \subseteq W[1] \subseteq W[2]$. It is widely believed that $\text{FPT} \neq W[1]$. In fact $\text{FPT} = W[1]$ implies that ETH fails. The Projection game conjecture (PGC) [11] is as follows.
Conjecture 2. There exists a constant $c$ so that for every $\epsilon > 1/n^c$ there is a PCP of almost linear size, namely, size $|I| \cdot 2^{|I|} \cdot \text{poly}(1/\epsilon)$ for some constant $0 < \alpha < 1$, with soundness $\epsilon$ and alphabet size $\text{poly}(1/\epsilon)$.

The One-Round 2 prover resulting from this PCP (and from all other PCP in the paper) obeys the projection property (see [11] for an explanation) and this is used in the reduction from PCP to SETCOVER (see [10]).

This conjecture is known to be valid, if alphabet has size $\exp(1/\epsilon)$. See [12].

We note that in [11] the sublinear term is not specified. Hence we did the natural thing and took the sublinear term from [12].

Theorem 1. [2] Under ETH and PGCC, there exist constants $1 > F_1, F_2 > 0$ such that the SETCOVER problem does not admit an FPT approximation algorithm with ratio $\text{OPT}^{F_1}$ in time $2^{\text{OPT}^{F_2}} \cdot \text{poly}(n)$.

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

In both results the time is strictly subexponential in $\text{OPT}$.

4 Preliminaries

Impagliazzo et al. [7] formulated the following conjecture which is known as ETH. We assume ETH in all hardness results in this paper.

<table>
<thead>
<tr>
<th>Exponential Time Hypothesis (ETH)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$3$-SAT cannot be solved in $2^{\Theta(q+m)}$ 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 ETH, $3$-SAT cannot be solved in $2^{\Theta(m)(q+m)}$ time where $q$ is the number of variables and $m$ is the number of clauses.

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.

The following is a very complex corollary that is derived from [11], [8] and [10]. Because of space limitations, we do not provide the proof of this corollary here. In the full version we will sketch how is this corollary derived. In the corollary, for some constant $c$, $\epsilon$ can be chosen as any value so that $\epsilon \geq 1/m^c$ [11] with $m$ the number of clauses the reduction starts with. The soundness in [11] is $\epsilon$ but due to the reduction in [8] the gap between a yes and a no instance in SETCOVER shrinks to $\Theta(1/\sqrt{\epsilon})$ (namely, size of the yes instance for SETCOVER over the size of the no instance in SETCOVER is $1/\Theta(\sqrt{\epsilon})$).
Corollary 1. [11],[8],[10] There exist two polynomials \( P_1(m) \) and \( P_2(m) \) so that 
\[ P_2(m) = o(P_1(m)) \] and for every \( \epsilon \geq 1/m^2 \), there exists a reduction from 3-SAT to setcover with \( m \) clauses so that
1. The number of sets is \( m \cdot 2^{\log m} \cdot P_1(1/\epsilon) \) where \( \alpha \) a constant that satisfies 
   \[ 0 < \alpha < 1 \].
2. The number of elements is \( \text{poly}(m) \cdot \exp(\sqrt{1/\epsilon}) \).
3. The optimum for a yes instance is exactly 
   \[ \text{opt}_y = m \cdot 2^{\log m} \cdot P_2(1/\epsilon) \].
4. The optimum \( \text{opt}_n \) for a “no” instance is at least \( d \cdot \sqrt{1/\epsilon} \cdot \text{opt}_y \) for some constant \( d \).

Remark: Clearly, \( \text{opt}_y \) has to be significantly smaller than the number of all sets. 
The difference is because in \( \text{opt}_y \), one vertex (answer) is taken out of any query. 
By the size of the alphabet, the number of answers per query is polynomial in \( 1/\epsilon \). Thus, the relation of the total number of sets and \( \text{opt}_y \) is polynomial. Here we use the PGC, and in particular the assumption on the size of the alphabet that is the heart of the PGC.

5 Our results

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

Theorem 3. Under the ETH and PGC conjectures, SETCOVER is \((r,t)\)-FPT-hard for \( r(\text{opt}) = (\log \text{opt})^\gamma \) and \( t(\text{opt}) = \exp(\exp((\log \text{opt})^\gamma)) \cdot \text{poly}(n) = \exp(\text{opt}^{(\log \text{opt})^\gamma}) \cdot \text{poly}(n) \) 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 SETCOVER [4]. We prove a stronger hardness as \( \gamma > 1 \). However, clearly \( r \) is much smaller than in [2].

Moshkovitz conjectured in a private that there may be an almost linear PCP of size \( m \cdot \text{poly}(\log(m)) \cdot \text{poly}(1/\epsilon) \) with alphabet \( \text{poly}(1/\epsilon) \) with soundness \( \epsilon \).

Theorem 4. Under ETH and the above stronger version of PGC there exists constants \( d', d'' \), so that SETCOVER is \((r,t)\)-FPT-hard for \( r(\text{opt}) = \text{opt}^{d''} \) and \( t(\text{opt}) = \exp(\exp(\text{opt}^{d''})) \cdot \text{poly}(n) \).

Note that the running times in this result is almost doubly exponential in \( \text{opt} \), and it can be seen that this is essentially the best we can get under current knowledge. The fact that we were unable to prove Fellow’s conjecture is not related to our efforts but to the current state of knowledge. Proving the conjecture in full requires a parameterized version of the PCP.

We can also prove an inapproximability with super-exponential time in \( \text{opt} \) 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 \( \exp(\text{opt}^{\log \text{opt}})/\text{poly}(n) \) 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 [5]. In [5] 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 \(\text{exp}(\sqrt{f(n)})\). Among other things, this implies that 3-SAT can be solved in time roughly \(\text{exp}(\sqrt{n})\), which contradicts the ETH. Hence the the assumption in [5] is weaker and implies our assumption, namely the ETH.

Theorem 6 works for any \(\text{opt}\) and \(\text{opt} \leq \log n\) in particular, and thus improves the paper of Feige et al [5] in two ways. First we prove \(1/(1 - \epsilon)\)-hardness which for such small values of \(\text{opt}\) might be significantly harder than ruling out an exact solution. Second, as \(n = m^{1+o(1)}\) the \(r(\text{opt})\)-hardness holds even if we allow time \(2^{o(n)}\) time. For this we need just to properly adjust the \(o(m)\) term so that the time will indeed by \(2^{o(n)}\). It may seems 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 \(2^{o(n)}\). This time, \(2^{o(n)}\) strongly improves the time \(n^{\log n}\) of [5].

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 under the ETH, clique is \((r, t)\)-FPT-hard. for any function \(t\), however huge.

As a function of \(n\), the running time is \(2^{n^{1/Q(n)}}\) for an arbitrarily slowly growing \(Q(n)\). Thus Theorem 7 improves [5] in the same two ways as Theorem 6.

Theorem 8. Under the ETH, mmis is \((r, t)\)-FPT-hard in \(\text{opt}\) (and thus in \(k\) since \(\text{opt}\) is known) for any non-decreasing functions \(r\) and \(t\).

6 Inapproximability for Set Cover with super-exponential time in \(\text{OPT}\)

In this section we prove Theorem 3.

Let \(P_1(m)\) and \(P_2(m)\) be two polynomial as in Corollary 1. We use \(\epsilon = c'/\log^2 m\) with a large enough constant \(c'\). A large enough constant \(c'\) with assure a gap that is at least \(\log m\) because \(1/\sqrt{c} = \Theta(\log m)\). By plugging \(\epsilon\) in Corollary 1 we get:

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 \(\text{PGC}\) and ETH, there exists a reduction from 3-SAT to setcover so that the number of sets in the resulting instance is \(\sigma = m \cdot 2^{\log^c m \cdot P_1((\log m))}\). Furthermore, value of the optimum in yes instance is exactly \(\text{OPT}_y = m \cdot 2^{\log^c m \cdot P_2((\log m))}\) and that in the no instance is at least \(\text{OPT}_y \geq \log m \cdot \text{OPT}_y\).
We now describe a way to change the \textsc{setcover}. The idea is to make the optimum much smaller. Starting with the \textsc{setcover} instance $S = (U, S)$ in the above corollary, where $U$ is the set of elements and $S \subseteq 2^{U}$ is the collection of sets, we construct a new instance $S' = (U, S')$ on the same elements as follows. We introduce a set $s \in S'$ as $s = \bigcup_{i=1}^{p} s_i$ for each subcollection $\{s_1, s_2, \ldots, s_p\} \subseteq S$ of size $p$ where $1 \leq p \leq \lfloor m / \log m \rfloor$.

**Claim.** The number of sets in the new instance $S' = (U, S')$ 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 $\sigma = m \cdot 2^{\log \alpha m} \cdot P_2((\log(m)))$ because of the choice of $\epsilon$. Thus since $p \leq \sigma / 2$, the number of sets in the new instance is

$$\sum_{p=1}^{\lfloor m / \log m \rfloor} \binom{\sigma}{p} \leq (m / \log m) \cdot (m \cdot 2^{\log \alpha m} \cdot P_2((\log(m))) \cdot (m / \log m)^{\lfloor m / \log m \rfloor} = 2^{o(m)}.$$  

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

$$\frac{(m / \log m) \cdot (m \cdot 2^{\log \alpha m} \cdot P_2((\log(m))) \cdot 2 \log m)^{m / \log m}}{2 \log m} = 2^{O(m / \log^{1-\alpha} m)} = 2^{o(m)}.$$  

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.

**Remark:** here we can see that if the PCP would not have had almost linear size, calculations will lead to running time would be larger than $2^{o(m)}$.

**Proof of Theorem 3.**

Clearly, any optimum will use as few sets of size (roughly) $m / \log m$ and so the gap between a “Yes” instance and a “No” hardly 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 optima of a yes instance and a no instance continues to be $\log m$ and the new optimum of the yes instance is at most $\text{OPT}_y = \lceil \text{OPT}_y / \lfloor m / \log m \rfloor \rceil = O(2^{\log \alpha m} \cdot P_2((\log(m)))$, and the optimum for a no instance $\text{OPT}_n$ is $\log m$ 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 $r(\text{OPT}_y') = O((\log^* m)^\gamma) = o((\log^* m)^{1/\alpha}) = o(\log m)$ and $t(\text{OPT}_n') = 2^{o(m)}$. Thus \textsc{setcover} is $(r, t)$-\text{FPT}-hard for these functions, proving Theorem 3.

6.1 A doubly exponential time inapproximability for \textsc{setcover}

Due to space limitation the proof is given in section in the appendix.

6.2 An inapproximability under the Exponential Time Hypothesis only

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

Theorem 9. There exist a positive $\epsilon > 0$ and a 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 $f(m) = 0(m)$ 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 $(x \lor \neg x \lor z_1), (x \lor \neg x \lor z_2), \ldots$ 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 $2^{o(m)}$.

Proof. Using $\binom{m}{k} \leq (ne/k)^k$, we get that the number of supervertices is at most $(7e \cdot f(m))^{m/f(m)} = \exp(\log(7e \cdot f(m)) \cdot 7m/f(m)) = 2^{o(m)}$, 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 $2^{o(m)}$.

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 $A \cap B = \emptyset$. 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 $h(\text{opt}) = 2^{O(m)}$. Selecting $f(m) = t^{-1}(h(m))$ gives $h(m) = 2^{o(m)}$ time. Since $m, n$ are linearly related here the time can be set to $2^{o(n)}$ for any $t$.

7.1 A super constant inapproximability

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

8 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, $W(C) = \{w^1_C, \ldots, w^q_C\}$.

Intuitively, we want to create a 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 SETCOVER and 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 $S \subseteq \{u_x \mid x \in C\} \cup \{\bar{u}_x \mid x \in C\}$ 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 $C = (x \lor \bar{x} \lor w)$. 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 $2^{o(q)} + qm$. The instance $G(I)$ can be constructed in time $2^{o(q)}$.

**Proof.** The total number of vertices in $H(I)$ of type $v_S$ for $S \subset A$ is at most $(\frac{q}{f(q)})^n < (qe/(q/f(q)))^{n/f(q)} < 2^{o(q)}$. Here we again use the inequality $(\frac{q}{k}) \leq (qe/k)^k$. 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 $v_S$, 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 $\{v_{T_i}\}$ defined above form a dominating set in $H(I)$.

**Proof.** We first show each vertex in $W(C)$ is adjacent to some vertex $v_{T_i}$. 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 fours cases above, by definition, there is an edge between $v_{T_i}$ and $v_S$. 

Thus we just proved the following corollary.

**Corollary 3.** The “yes” instance admits a solution of size $f(q)$.

**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 $v_{\bar{T}}$ 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 $t(f(q)) = 2^{o(q)}$. Note that by Claims 8 and 8, 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-SAT in time $2^{o(q)}$, contradicting the ETH.

**References**

A Proof of Theorem 4

Let $P_1$ and $P_2$ be polynomial as in Corollary 1. For proving this theorem we assume:

Conjecture 3. There exists a constant $c > 0$ and a PCP of size $m \cdot \text{poly\,log}(m) P_1(1/\epsilon)$, for any $\epsilon$ so that $\epsilon \geq 1/m^c$.

This result was conjectured to hold by Moshkovitz in a private communication. 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 $\epsilon = c'/\log^2 m$ for a large enough constant $c'$ we get that:

Corollary 4. There exists a a reduction from 3-SAT to SETCOVER so that:

1. The number of sets is $\sigma = m \cdot P_1(\log(m))$
2. The number of elements is $\text{poly}(m)$.
3. The value of the optimum in yes instance is exactly $\text{opt}_y = m \cdot P_2((\log(m)))$ and that in the “no” instance is larger by a factor of at least $\log m$.

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:

$$\left(\frac{m \cdot P_2((\log(m)))}{d \log \log m}\right)^m$$

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 $2^{\log^a m}$ is gone.

After this change, the size of the optimum for a “yes” instance becomes $\text{opt}' = P_2((\log(m)))$. Recall that the gap is $\log m$. Therefore, the gap can be stated as $\text{opt}^d$ for some $d < 1$.

We choose $d'$ so that $\text{opt}^d = \log m$, and get that for every $d'' < d'$ $2^{\text{opt}^{d''}} = o(m)$ and the running time is $\exp(2^{\text{opt}^{d''}}) = 2^{o(m)}$. This ends the proof of Theorem 4.

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:

$$2^{2 \log(7 - e \cdot f(m))} 7m/f(m).$$

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 3. 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 if 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 $(1 - \epsilon)\sqrt{f(\text{OPT})}$, $\text{OPT}\sqrt{f(\text{OPT})}$.

Note that the new size of the graph is:

$$2^{2^{\sqrt{f(m)} \log(7 - f(m))} \frac{7m}{f(m)}} = 2^{o(m)}.$$

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 7. 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 $n = n'/\sqrt{f(m)}$. 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 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 [12]. Let $P_1(m)$ and $P_2(m)$ be two polynomials as in Corollary 1.
Theorem 12. There exists a constant $c$ and a PCP of size $m \cdot 2^{\log^\alpha m} \cdot \text{poly}(1/\epsilon)$, 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 answers per query a lot, and so increases the size of the reduced instance. Indeed, the number of answers is now

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

Thus this is the size of the Labelcover graph.

We choose $\epsilon = \ln 2 \cdot \log \alpha m$. 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 $m \cdot 2^{\log^\alpha m}$
2. The number of elements is $\text{poly}(m)$.
3. The gap is $d \cdot \sqrt{\log \alpha m}$.
4. The optimum of a yes instance does not change, namely, is $\text{OPT} = 2^{\log^\alpha m} \cdot \text{poly log}(m)$.

The proofs here are simple computations using the new value of $\epsilon$ plugged in Corollary 1. The optimum $\text{OPT}_y$ 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 queries.

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

The time in terms of $\text{opt}$: Since $\text{OPT}_y$ 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.