On Subexponential and FPT-time Inapproximability*

Edouard Bonnet, Bruno Escoffier, Eun Jung Kim, and Vangelis Th. Paschos**

PSL Research University, Université Paris-Dauphine, LAMSADE, CNRS UMR 7243
{escoffier,eun-jung.kim,paschos}@lamsade.dauphine.fr

Abstract. Fixed-parameter algorithms, approximation algorithms and moderately exponential algorithms are three major approaches to algorithms design. While each of them being very active in its own, there is an increasing attention to the connection between these different frameworks. In particular, whether Independent Set would be better approximable once endowed with subexponential-time or FPT-time is a central question. In this article, we provide new insights to this question using two complementary approaches; the former makes a strong link between the linear PCP conjecture and inapproximability; the latter builds a class of equivalent problems under approximation in subexponential time.

1 Introduction

Fixed-parameter algorithms, approximation algorithms and moderately exponential/subexponential algorithms are major approaches for efficiently solving NP-hard problems. These three areas, each of them being very active in its own, have been considered as foreign to each other until recently. Polynomial-time approximation algorithm produces a solution whose quality is guaranteed to lie within a certain range from the optimum. One illustrative problem indicating the development of this area is Independent Set. The approximability of Independent Set within constant ratios has remained as the most important open problems for a long time in the field. It was only after the novel characterization of the **NP** by PCP theorem [1] that such inapproximability was proven assuming $\mathbf{P} \neq \mathbf{NP}$. Subsequent improvements of the original PCP theorem led to much stronger result for Independent Set: it is inapproximable within ratios $\Omega(n^{\varepsilon-1})$ for any $\varepsilon > 0$, unless $\mathbf{P} = \mathbf{NP}$ [2].

The design of moderately exponential (subexponential, respectively) algorithms allows exponential (subexponential, respectively) running time for the sake of optimality. In this case, the endeavor lies in making the growth of the running time function as slow as possible. Parameterized complexity provides an alternative framework to analyze the running time in a more refined way [3]. Given an instance with a parameter k, the aim is to get an $O(f(k) \cdot n^c)$-time (or

* Research supported by the French Agency for Research under the DEFIS program TODO, ANR-09-EMER-010

** Also, Institut Universitaire de France equivalently, FPT-time) algorithm for some constant c, where the constant c is independent of k. As these two research programs offer a generous running time when compared to that of classic approximation algorithms, a growing amount of attention is paid to them as a way to cope with hardness in approximability. The first one yields *moderately exponential approximation*. In moderately exponential approximation, the core question is whether a problem is approximable in moderately exponential time while such approximation is impossible in polynomial time. Suppose a problem is solvable in time $O^*(\gamma^n)$, but it is NP-hard to approximate within ratio r. Then, we seek for r-approximation algorithms of running time significantly faster than $O^*(\gamma^n)$. This issue has been considered for several problems [4–6, 12, 16].

The second research program handles approximation by fixed parameter algorithms. We say that a minimization (maximization, respecitvely) problem Π, together with a parameter k, is *parameterized r-approximable* if there exists an FPT-time algorithm which computes a solution of size at most (at least, respectively) rk whenever the input instance has a solution of size at most (at least, respectively) k. This line of research was initiated by three independent works [14, 8, 10]. For an excellent overview, see [21]. In what follows, parameterization means "standard parameterization", i.e., where problems are parameterized by the cost of the optimal solution.

Several natural questions can be asked dealing with these two programs. In particular, the following ones have been asked several times [21, 14, 16, 6].
Q1: can a problem, which is highly inapproximable in polynomial time, be well-approximated in subexponential time?
Q2: does a problem, which is highly inapproximable in polynomial time, become well-approximable in FPT-time?

Few answers have been obtained until now. Regarding **Q1**, negative results can be directly obtained by gap-reductions for certain problems. For instance, Coloring is not approximable within ratio $4/3 - \epsilon$ since this would allow to determine whether a graph is 3-colorable or not in subexponential time. This contradicts a widely-acknowledge computational assumption [18]:

> *Exponential Time Hypothesis* (ETH): There exists an $\epsilon > 0$ such that no algorithm solves 3Sat in time $2^{\epsilon n}$, where n is the number of variables.

Regarding **Q2**, [14] shows that assuming FPT $\neq$ W[2], for any r the Independent Dominating Set problem is not r-approximable[1] in FPT-time.

Among interesting problems for which **Q1** and **Q2** are worth being asked are Independent Set, Coloring and Dominating Set. They fit in the frame of both **Q1** and **Q2** above: they are hard to approximate in polynomial time while their approximability in subexponential or in parameterized time is still open.

In this paper, we study parameterized and subexponential (in)approximability of natural optimization problems. In particular, we follow two guidelines: (i) getting inapproximability results under some conjecture and (ii) establishing

[1] Actually, the result is even stronger: it is impossible to obtain a ratio $r = g(k)$ for any function g.

classes of uniformly inapproximable problems under approximability preserving reductions.

Following the first direction, we establish a link between a major conjecture in PCP theorem and inapproximability in subexponential-time and in FPT-time, assuming ETH. Just below, we state this conjecture while the definition of PCP is deferred to the next section.

> *Linear PCP Conjecture* (LPC): 3Sat $\in \text{PCP}_{1,\beta}[\log|\phi| + D, E]$ for some $\beta \in (0,1)$, where $|\phi|$ is the size of the 3Sat instance (sum of lengths of clauses), D and E are constant.

Unlike ETH which is conjectured to be true, LPC is a wide open question. In Lemma 1 stated in Section 2, we claim that if LPC turns out to hold, it implies that one of the most interesting questions in subexponential and parameterized approximation is answered in the negative. In particular, the following hold for Independent Set on n vertices, for *any* constant $0 < r < 1$ assuming ETH:
(i) There is no r-approximation algorithm in time $O(2^{n^{1-\delta}})$ for any $\delta > 0$.
(ii) There is no r-approximation algorithm in time $O(2^{o(n)})$, if LPC holds.
(iii) There is no r-approximation algorithm in time $O(f(k)n^{O(1)})$, if LPC holds.

Let us note that (i) is not conditional upon LPC. In fact, this is an immediate consequence of near-linear PCP construction achieved in [13]. Note that similar inapproximability results under ETH for Max-3Sat and Max-3Lin for some subexponential running time have been obtained in [23].

Following the second direction, we show that a number of problems are equivalent with respect to approximability in subexponential time. Designing a family of equivalent problems is a common way to provide an evidence in favor of hardness of these problems. One prominent example is the family of problems complete under SERF-reducibility [18] which leads to equivalent formulations of ETH. More precisely, for a given problem Π, let us formulate the following hypothesis, which can be seen as the approximate counterpart of ETH.

Hypothesis 1 (APETH(Π)) *There exist two constants $\epsilon > 0$ and r ($r < 1$ if Π is a maximization problem, $r > 1$, otherwise), such that Π is not r-approximable in time $O(2^{\epsilon n})$.*

We prove that several well-known problems are equivalent with respect to the APETH (APETH-equivalent), in the sense that verification or not of the APETH by one of them, implies the same fact for the other ones. To this end, a notion called the *approximation preserving sparsification* is proposed. A recipe to prove that two problems A and B are APETH-equivalent consists of two steps. The first is to reduce an instance of A into a family of instances in "bounded" version (bounded degree for graph problems, bounded occurrence for satisfiability problems), which are equivalent with respect to approximability. This step is where the proposed notion comes into play. The second is to use standard approximability preserving reductions to derive equivalences between bounded versions of A and B. In this paper, we consider L-reductions [24] for this purpose. Furthermore, we show that if APETH fails for one of these problems, then *any* problem in MaxSNP would be approximable for *any* constant ratio in subexponential FPT-time $2^{o(k)}$. This result can be viewed as an extension of [9], which states that none of MaxSNP hard problems allows $2^{o(k)}$-time algorithm under ETH. Also, it could be considered as an evidence toward the validity of APETH

Some preliminaries and notation are given in Section 2. Results derived from PCP and LPC are given in Section 3. The second direction on equivalences between problems is described in Section 4.

2 Preliminaries and notation

We denote by $\mathrm{PCP}_{\alpha,\beta}[q,p]$ (see for instance [1] for more on PCP systems) the set of problems for which there exists a PCP verifier which uses q random bits, reads at most p bits in the proof and is such that: (1) if the instance is positive, then there exists a proof such that V(erifier) accepts with probability at least α; (2) if the instance is negative, then for any proof V accepts with probability at most β. The following theorem is proved in [13] (see also Theorem 7 in [23]), presenting a further refinement of the characterization of NP.

Theorem 1. *[13] For every $\epsilon > 0$,*

$$3\textsc{Sat} \in \mathrm{PCP}_{1,\epsilon}[(1+o(1))\log n + O(\log(1/\epsilon)), O(\log(1/\epsilon))]$$

A recent improvement [23] of Theorem 1 (a PCP Theorem with two-query projection tests, sub-constant error and almost-linear size) has some important corollaries in polynomial approximation. In particular:

Corollary 1. *[23] Under* ETH*, for every $\epsilon > 0$, and $\delta > 0$, it is impossible to distinguish between instances of* Max-3Sat *with m clauses where at least $(1-\epsilon)m$ are satisfiable from instances where at most $(7/8+\epsilon)m$ are satisfiable, in time $O(2^{m^{1-\delta}})$.*

Under LPC, a stronger version of this result follows from standard argument[2].

Lemma 1. *If* L*PC*[3] *and ETH hold, then there exists $r < 1$ such that for every $\epsilon > 0$ it is impossible to distinguish between instances of* Max-3Sat *with m clauses where at least $(1-\epsilon)m$ are satisfiable from instances where at most $(r+\epsilon)m$ are satisfiable, in time $2^{o(m)}$.*

This (conditional) hardness result of approximating Max-3Sat will be the basis of the negative results of parameterized approximation in Section 3.1.

Let us now present two useful gap amplification results for Independent Set. First, as noted in [15], the so-called self-improvement property [17] can be proven for Independent Set also in the case of parameterized approximation.

[2] All missing proofs can be found in appendix.

[3] Note that LPC as expressed here implies validity of the lemma even with replacing $(1-\epsilon)m$ by m. However, we stick with this lighter statement $(1-\epsilon)m$ in order, in particular, to emphasize the fact that perfect completeness is not required in the LPC conjecture.

Lemma 2. *[15] If there exists a parameterized r-approximation algorithm for some $r \in (0,1)$ for* Independent Set*, then this is true for any $r \in (0,1)$.*

It is also well known that the very powerful tool of expander graphs allows to derive the following gap amplification for Independent Set (see Appendix A).

Theorem 2. *Let G be a graph on n vertices (for a sufficiently large n) and $a > b$ be two positive real numbers. Then for any real $r > 0$ one can build in polynomial time a graph G_r and specify constants a_r and b_r such that: (i) G_r has $N \leqslant Cn$ vertices, where C is some constant independent of G (but may depend on r); (ii) if $\omega(G) \leqslant bn$ then $\omega(G_r) \leqslant b_r N$; (iii) if $\omega(G) \geqslant an$ then $\omega(G_r) \geqslant a_r N$; (iv) $b_r/a_r \leqslant r$.*

Finally, we will use in the sequel the well known sparsification lemma [18]. Intuitively, this lemma allows to work with 3-SAT formula with linear lengths (the sum of the lengths of clauses is linearly bounded in the number of variables).

Lemma 3. *[18] For all $\epsilon > 0$, a 3-SAT formula ϕ on n variables can be written as the disjunction of at most $2^{\epsilon n}$ 3-SAT formulæ ϕ_i on (at most) n variables such that ϕ_i contains each variable in at most c_ϵ clauses for some constant c_ϵ. Moreover, this reduction takes at most $p(n)2^{\epsilon n}$ time.*

3 Some consequences of (almost-)linear size PCP system

3.1 Parameterized inapproximability bounds

It is shown in [11] that, under ETH, for any function f no algorithm running in time $f(k)n^{o(k)}$ can determine whether there exists an independent set of size k, or not (in a graph with n vertices). A challenging question is to obtain a similar result for approximation algorithms for Independent Set. In the sequel, we propose a reduction from Max-3Sat to Independent Set that, based upon the negative result of Corollary 1, only gives a negative result for *some* function f (because Corollary 1 only avoids *some* subexponential running times). However, this reduction gives the inapproximability result sought, if the consequence of LPC given in Lemma 1 (which strengthens Corollary 1 and seems to be a much weaker assumption than LPC) is used instead. We emphasize the fact that the results in this section are valid as soon as a hardness result for Max-3Sat as that in Lemma 1 holds.

The proof of the following theorem essentially combines the parameterized reduction in [11] and a classic gap-preserving reduction.

Theorem 3. *Under* LPC *and* ETH*, there exists $r < 1$ such that, no fixed parameter approximation algorithm for* Independent Set *running in time $f(k)n^{o(k)}$ can achieve approximation ratio r in graphs of order n.*

The following result follows from Lemma 2 and Theorem 3.

Corollary 2. *Under* LPC *and* ETH*, for any $r \in (0,1)$ there is no r-approximation parameterized algorithm for* Independent Set *(i.e., an algorithm that runs in time $f(k)p(n)$ for some function f and some polynomial p).*

Let us now consider Dominating Set which is known to be W[2]-hard [3]. The existence of parameterized approximation algorithms for this problem is open [14]. Here, we present an approximation preserving reduction (fitting the parameterized framework) which, given a graph $G(V, E)$ on n vertices where V is a set of K cliques $C_1, \cdots, C_K$, builds a graph $G'(V', E')$ such that G has an independent set of size α if and only if G' has a dominating set of size $2K - \alpha$. Using the fact that the graphs produced in the proof of Theorem 3 are of this form (vertex set partitioned into cliques), this reduction will allow us to obtain a lower bound (based on the same hypothesis) for the approximation of min dominating set from Theorem 3.

The graph G' is built as follows. For each clique C_i in G, add a clique C'_i of the same size in G'. Add also: an independent set S_i of size $3K$, each vertex in S_i being adjacent to all vertices in C'_i and a special vertex t_i adjacent to all the vertices in C'_i. For each edge $e = (u, v)$ with u and v *not* in the same clique in G, add an independent set W_e of size $3K$. Suppose that $u \in C_i$ and $v \in C_j$. Then, each vertex in W_e is linked to t_i and to all vertices in C'_i but u, and t_j and all vertices in C'_j but v.

Informally, the reduction works as follows. The set S_i ensures that we have to take at least one vertex in each C'_i, the fact that $|W_e| = 3K$ ensures that it is never interesting to take a vertex in W_e. If we take t_i in a dominating set, this will mean that we do not take any vertex in the set C_i in the corresponding independent set in G. If we take one vertex in C'_i (but not t_i), this vertex will be in the independent set in G. Let us state this property in the following lemma.

Lemma 4. *G has an independent set of size α if and only if G' has a dominating set of size $2K - \alpha$.*

Theorem 4. *Under LPC and ETH, there exists an $r > 1$ such that there is no r-approximation algorithm for* Dominating Set *running in time $f(k)n^{o(k)}$ where n is the order of the graph.*

Such a lower bound immediately transfers to Set Cover since a graph on n vertices for Dominating Set can be easily transformed into an equivalent instance of Set Cover with ground set and set system both of size n.

Corollary 3. *Under LPC and ETH, there exists $r > 1$ such that there is no r-approximation algorithm for* Set Cover *running in time $f(k)m^{o(k)}$ in instances with m sets.*

3.2 On the approximability of Independent Set and related problems in subexponential time

As mentioned in Section 2, an almost-linear size PCP construction [23] for 3Sat allows to get the negative result stated in Corollary 1. In this section, we present further consequences of Theorem 1, based upon a combination of known reductions with (almost) linear size amplifications of the instance.

First, Theorem 1 combined with the reduction in [1] showing inapproximability results for Independent Set in polynomial time and the gap amplification of Theorem 2, leads to the following result.

Theorem 5. *Under* ETH*, for any $r > 0$ and any $\delta > 0$, there is no r-approximation algorithm for* Independent Set *running in time $O(2^{n^{1-\delta}})$, where n is the order of the input graph.*

Since (for $k \leqslant n$), $n^{k^{1-\delta}} = O(2^{n^{1-\delta'}})$, for some $\delta' < \delta$, the following holds.

Corollary 4. *Under* ETH*, for any $r > 0$ and any $\delta > 0$, there is no r-approximation algorithm for* Independent Set *(parameterized by k) running in time $O(n^{k^{1-\delta}})$, where n is the order of the input graph.*

The results of Theorem 5 and Corollary 4 can be immediately extended to problems that are linked to Independent Set by approximability preserving reductions (that preserve at least constant ratios) that have linear amplifications of the sizes of the instances, as in the following proposition.

Proposition 1. *Under* ETH*, for any $r > 0$ and any $\delta > 0$, there is no r-approximation algorithm for either* Set Packing *or* Bipartite Subgraph *running in time $O(2^{n^{1-\delta}})$ in a graph of order n.*

Dealing with minimization problems, Theorem 5 and Corollary 4 can be extended to Coloring, using the reduction given in [20]. Note that this reduction uses the particular structure of graphs produced in the inapproximability result in [1] (as in Theorem 5). Hence, the following result can be derived.

Proposition 2. *Under* ETH*, for any $r > 1$ and any $\delta > 0$, there is no r-approximation algorithm for* Coloring *running in time $O(2^{n^{1-\delta}})$ in a graph of order n.*

Concerning the approximability of Vertex Cover and Min-Sat in subexponential time, the following holds.

Proposition 3. *Under* ETH*, for any $\epsilon > 0$ and any $\delta > 0$, there is no $(7/6 - \epsilon)$-approximation algorithm for* Vertex Cover *running in time $O(2^{n^{1-\delta}})$ in graphs of order n, nor for* Min-Sat *running in time $2^{m^{1-\delta}}$ in CNF formulæ with m clauses.*

All the results given in this section are valid under ETH and rule out some ratios in subexponential time of the form $2^{n^{1-\delta}}$. It is worth noticing that if LPC holds, then all these results would hold for *any* subexponential time. Note that this is in some sense optimal since it is easy to see that, for any increasing and unbounded function $r(n)$, Independent Set is approximable within ratio $1/r(n)$ in subexponential time (simply consider all the subsets of V of size at most $n/r(n)$ and return the largest independent set among these sets).

Corollary 5. *If* L*PC holds, under* ETH *the negative results of Theorem 5 and Propositions 1, 2 and 3 hold for any time complexity $2^{o(n)}$.*

4 Subexponential approximation preserving reducibility

In this section, we study subexponential approximation preserving reducibility. Recall that APETH(Π) (Hypothesis 1) states that it is hard to approximate in subexponential time problem Π, within some constant ratio r. We exhibit that a set of problems are APETH-equivalent using the notion of *approximation preserving sparsification*. We then link APETH with approximation in subexponential FPT-time.

4.1 Approximation preserving sparsification and APETH equivalences

Recall that the sparsification lemma for 3SAT reduces a formula ϕ to a set of formulae ϕ_i with bounded occurrences of variables such that solving the instances ϕ_i would allow to solve ϕ. We attempt to build an analogous construction for subexponential approximation using the notion of *approximation preserving sparsification*. Given an optimization problem Π and some parameter of the instance, Π-B denotes the problem restricted to instances where the parameter is at most B. In what follows, we prescribe the maximum degree of a graph or the maximum number of literal occurrences as the parameter. Then Π-B would be the problems restricted to instances with the parameter bounded by B.

Definition 1. *An approximation preserving sparsification from a problem Π to a bounded parameter version Π-B of Π is a pair (f, g) of functions such that, given any $\epsilon > 0$ and any instance I of Π:*

1. *f maps I into a set $f(I, \epsilon) = (I_1, I_2, \ldots, I_t)$ of instances of Π, where $t \leqslant 2^{\epsilon n}$ and $n_i = |I_i| \leqslant n$; moreover, there exists a constant B_ϵ (independent on I) such that any I_i has parameter at most B_ϵ;*
2. *for any $i \leqslant t$, g maps a solution S_i of an instance I_i (in $f(I, \epsilon)$) into a solution S of I;*
3. *there exists an index $i \leqslant t$ such that if a solution S_i is an r-approximation in I_i, then $S = g(I, \epsilon, I_i, S_i)$ is an r-approximation in I;*
4. *f is computable in time $O^*(2^{\epsilon n})$, and g is polynomial with respect to $|I|$.*

With a slight abuse of notation, let us APETH(Π-B) denote the hypothesis: $\exists B$ such that APETH(Π-B), meaning that Π is hard to approximate in subexponential time even for some bounded parameter family of instances. Then the following holds[4].

Theorem 6. *If there exists an approximation preserving sparsification from Π to Π-B, then* APETH(Π) *if and only if* APETH(Π-B)*.*

[4] Note that we could consider a more general definition, leading to the same theorem, by allowing (1) a slight amplification of the size of I_i ($n_i \leqslant \alpha n$ for some fixed α in item 1), (2) an expansion of the ratio in item 3 (if S_i is r-approximate S is $h(r)$ approximate where $h(r)$ goes to one when r goes to one) and (3) a computation time $O^*(2^{\epsilon n})$ for g in item 4. But the simpler version of Definition 1 is sufficient to to guarantee validity of or results.

We now illustrate this technique on some problems. It is worth noticing that the sparsification lemma for 3Sat in [18] is *not* approximation preserving[5]; one cannot use it to argue that approximating Max-3Sat (in subexponential time) is equivalent to approximating Max-3Sat with bounded occurrences.

Proposition 4. *There exists an approximation preserving sparsification from* Independent Set *to* Independent Set-B *and one from* Vertex Cover *to* Vertex Cover-B.

Proof. Let $\epsilon > 0$. It is well known that the positive root of $1 = x^{-1} + x^{-1-B}$ goes to one when B goes to infinity. Then, consider a B_ϵ such that this root is at most 2^ϵ. Our sparsification is obtained via a branching tree: the leaves of this tree will be the set of instances I_i; f consists of building this tree; a solution of an instance in the leaf corresponds, via the branching path leading to this leaf, to a solution of the root instance, and that is what g makes.

More precisely, for Independent Set, consider the following usual branching tree, starting from the initial graph G: as long as the maximum degree is at least B_ϵ, consider a vertex v of degree at least B_ϵ, and branch on it: either take v in the independent set (and remove $N[v]$), or do not take it. The branching stops when the maximum degree of the graph induced by the unfixed vertices is at most $B_\epsilon - 1$. When branching, at least $B_\epsilon + 1$ vertices are removed when taking v, and one when not taking v; thus the number of leaves is $t \leqslant 2^\epsilon$ (by the choice of B_ϵ). Then, f and g satisfy items 1 and 2 of the definition. For item 3, it is sufficient to note that g maps S_i in S by adding adequate vertices. Then, if we consider the path in the tree corresponding to an optimal solution S^*, leading to a particular leaf G_i, we have that $|S^*| = |S^* \cap G_i| + k$ for some $k \geqslant 0$, and the solution S computed by g is of size $|S| = |S_i| + k$. So, $\frac{|S|}{|S^*|} \geqslant \frac{|S_i|}{|S^* \cap G_i|} \geqslant r$ if S_i is an r-approximation for G_i. The same argument holds also for Vertex Cover. □

Analogous arguments apply more generally to any problem where we have a "sufficiently good" branching rule when the parameter is large. Indeed, suppose we can ensure the decrease in instance size by $g(B)$ for nondecreasing and unbounded function g in all (possibly except for one) branches. Then such a branching rule can be utilized to yield an approximation preserving sparsification as in Proposition 4.

We give another approximation preserving sparsification, where there is no direct branching rule allowing to remove a sufficiently large number of vertices. Let Generalized Dominating Set be defined as follows: given a graph $G = (V, E)$ where V is partitioned into V_1, V_2, V_3, we ask for a minimum size set of vertices $V' \subseteq V_1 \cup V_2$ which dominates all vertices in $V_2 \cup V_3$. Of course, the case $V_2 = V$ corresponds to the usual Dominating Set problem. Note that Generalized Dominating Set is also a generalization of Set Cover, with $V_2 = \emptyset$, V_3 being the ground set and V_1 being the set system.

[5] One of the reasons is that when a clause C is contained in a clause C', a reduction rule removes C', that is safe for the satisfiability of the formula, but not when considering approximation.

Proposition 5. *There exists an approximation preserving sparsification from* Generalized Dominating Set *to* Generalized Dominating Set-B*.*

Combining Proposition 5 with some reductions, the following can be shown.

Lemma 5. APETH*(*Dominating Set*) implies* APETH*(*Independent Set-B*).*

Note that similarly, APETH(Set Cover) implies APETH(Independent Set-B), when the complexity of Set Cover is measured by $n + m$. Then, we have the following set of equivalent problems.

Theorem 7. Set Cover*,* Independent Set*,* Independent Set-B*,* Vertex Cover*,* Vertex Cover-B*,* Dominating Set*,* Dominating Set-B*,* Max Cut-B*,* 3Sat-B*,* Max-kSat-B *(for any $k \geqslant 2$) are* APETH*-equivalent.*

Proof. The equivalences between Vertex Cover-B, Independent Set-B, Max Cut-B, 3Sat-B, Max-2Sat-B, Dominating Set-B follow immediately from [24]. Indeed, for these problems [24] provides L-reductions with linear size amplification. The equivalence between Max-kSat-B problems is also well known (just replace a clause of size k by $k - 1$ clauses of size 3).

The equivalence between Independent Set and Independent Set-B, Vertex Cover and Vertex Cover-B follows from Proposition 4. Finally, Lemma 5 allows us to conclude for Dominating Set.

4.2 APETH and parameterized approximation

The equivalence drawn in Theorem 7 gives a first intuition that the corresponding problems should be hard to approximate in subexponential time for some ratio. In this section we show another argument towards this hypothesis: if it fails, then *any* MaxSNP problem admits for *any* $r < 1$ a parameterized r-approximation algorithm in subexponential time $2^{o(k)}$, which would be quite surprising. The following theorem can be construed as an extension of [9].

Theorem 8. *The following statements are equivalent:*

(i) APETH(Π) holds for one (equivalently all) problem(s) in Theorem 7;
(ii) there exist a MaxSNP-complete problem Π, some ratio $r < 1$ and a constant $\epsilon > 0$ such that there is no parameterized r-approximation algorithm for Π with running time $O(2^{\epsilon k} poly(|I|))$;
(iii) for any MaxSNP-complete problem Π, there exist a ratio $r < 1$ and an $\epsilon > 0$ such that there is no parameterized r-approximation algorithm for Π with running time $O(2^{\epsilon k} poly(|I|))$.

As an interesting complement of the above theorem, we show that trade-offs between (exponential) running time and approximation ratio do exist for any MaxSNP problem. In [7], it is shown that every MaxSNP problem Π is fixed-parameter tractable in time $2^{O(k)}$ for the standard parameterization, while in [24] it is shown that Π is approximable in polynomial time within a constant ratio

ρ_Π. We prove here that there exists a family of parameterized approximation algorithms achieving ratio $\rho_\Pi + \epsilon$, for any $\epsilon > 0$, and running in time $2^{O(\epsilon k)}$. This is obtained as a consequence of a result in [19].

Proposition 6. *Let Π be a standard parameterization of a MaxSNP-complete problem. For any $\epsilon > 0$, there exists a parameterized $(\rho_\Pi + \epsilon)$-approximation algorithm for Π running in time $\gamma^{\epsilon k} \cdot poly(|I|)$ for some constant γ.*

5 Conclusion

More interesting questions remain untouched in the junction of approximation and (sub)exponential-time/FPT-time computations. This paper is only a first step in this direction and we wish to motivate further research. Among a range of problems to be tackled, we propose the followings.

- Our inapproximability results are conditional upon Linear PCP Conjecture. Is it possible to relax the condition to a more plausible one?
- Or, we dare ask whether (certain) inapproximability results in FPT-time imply strong improvement in PCP theorem. For example, would the converse of Lemma 1 hold?
- Can we design approximation preserving sparsifications for problems like Max Cut or Max-3Sat? It seems to be difficult to design a sparsifier based on branching rules, so a novel idea is needed.

Note that we have considered in this article constant approximation ratios. As noted earlier, ratio $1/r(n)$ is achievable in subexponential time for any increasing and unbounded function r. However, dealing with parameterized approximation algorithms, achieving a non-constant ratio is also an open question. More precisely, finding in FPT-time an independent set of size $g(k)$ when there exists an independent set of size k is not known for *any* unbounded and increasing function g.

Finally, let us note that, in the same vein of the first part of our work, Mathieson [22] recently studied a proof checking view of parameterized complexity. Possible links between these two approaches are worth being investigated in future works.

References

1. Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M.: Proof verification and intractability of approximation problems. J. ACM **45** (1998) 501–555
2. Zuckerman, D.: Linear degree extractors and the inapproximability of max clique and chromatic number. In Proc. STOC'06. (2006) 681–690
3. Downey, R.G., Fellows, M.R.: Parameterized complexity. Monographs in Computer Science. Springer, New York (1999)
4. Björklund, A., Husfeldt, T., Koivisto, M.: Set partitioning via inclusion-exclusion. SIAM J. Comput. **39** (2009) 546–563

5. Bourgeois, N., Escoffier, B., Paschos, V.T.: Efficient approximation of MIN COLORING by moderately exponential algorithms. Inform. Process. Lett. **109** (2009) 950–954
6. Bourgeois, N., Escoffier, B., Paschos, V.T.: Approximation of MAX INDEPENDENT SET, MIN VERTEX COVER and related problems by moderately exponential algorithms. Discrete Appl. Math. **159** (2011) 1954–1970
7. Cai, L., Chen, J.: On fixed-parameter tractability and approximability of np optimization problems. J. Comput. System Sci. **54** (1997) 465–474
8. Cai, L., Huang, X.: Fixed-parameter approximation: conceptual framework and approximability results. In Proc. IWPEC'06. LNCS 4169, 96–108
9. L. Cai and D. W. Juedes. On the existence of subexponential parameterized algorithms. *J. Comput. Syst. Sci.*, 67(4): 789-807, 2003.
10. Chen, Y., Grohe, M., Grüber, M.: On parameterized approximability. In Proc. IWPEC'06. LNCS 4169, 109–120
11. Chen, J., Huang, X., Kanj, I.A., Xia, G.: Strong computational lower bounds via parameterized complexity. J. Comput. System Sci. **72** (2006) 1346–1367
12. Cygan, M., Pilipczuk, M.: Exact and approximate bandwidth. Theoret. Comput. Sci. **411** (2010) 3701–3713
13. Dinur, I.: The PCP theorem by gap amplification. J. ACM **54** (2007)
14. Downey, R.G., Fellows, M.R., McCartin, C.: Parameterized approximation problems. In Proc. IWPEC'06. LNCS 4169, 121–129
15. Escoffier, B., Paschos, V.T., Tourniaire, E.: Moderately exponential and parameterized approximation: some structural results. (Unpublished manuscript)
16. Fürer, M., Gaspers, S., Kasiviswanathan, S.P.: An exponential time 2-approximation algorithm for bandwidth. In Proc. IWPEC'09, LNCS 5917, 173–184
17. Garey, M.R., Johnson, D.S.: Computers and intractability. A guide to the theory of NP-completeness. W. H. Freeman, San Francisco (1979)
18. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? J. Comput. System Sci. **63** (2001) 512–530
19. Kim, E.J., Williams, R.: Improved parameterized algorithms for above average constraint satisfaction. In Proc. IPEC'11. LNCS 7112, 118–131
20. Lund, C., Yannakakis, M.: On the hardness of approximating minimization problems. J. Assoc. Comput. Mach. **41** (1994) 960–981
21. Marx, D.: Parameterized complexity and approximation algorithms. The Computer Journal **51** (2008) 60–78
22. Mathieson, L.: A proof checking view of parameterized complexity. CoRR **abs/1206.2436** (2012)
23. Moshkovitz, D., Raz, R.: Two query pcp with sub-constant error. In: Proc. FOCS'08. (2008) 314–323
24. Papadimitriou, C.H., Yannakakis, M.: Optimization, approximation and complexity classes. J. Comput. System Sci. **43** (1991) 425–440
25. Vazirani, V.: Approximation algorithms. Springer, Berlin (2001)

A Gap amplification

Let us first recall the definition of an expander graph.

Definition 2. *A graph G is a (n, d, α)-expander graph if (i) G has n vertices, (ii) G is d-regular, (iii) all the eigenvalues λ of G but the largest one is such that $|\lambda| \leqslant \alpha d$.*

Fact 1. *For any $k \in \mathbb{N}^*$ and any $\alpha > 0$ there exists d and a (k^2, d, α)-expander graph. Moreover, d depends only on α, and this graph can be computed in polynomial time for every fixed α.*

This fact follows from the following lemmata.

Lemma 6. *(O. Gabber and Z. Galil, Explicit constructions of linear-sized superconcentrators, J. Comput. System Sci., 22(3):407–420, 1981, or Th. 8.1 in S. Hoory, N. Linial and A. Widgerson, Expander graphs and their applications, Bulletin of the AMS. 43 (4), 439–561, 2006)*
For every positive integer k, there exists a $(k^2, 8, 5\sqrt{2}/8)$-expander graph, computable in polynomial time.

If G is a graph with adjacency matrix M, let us denote G^k the graph with adjacency matrix M^k.

Lemma 7. *(Fact 1.2 in O. Reingold, S. P. Vadhan and A. Wigderson, Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors, Electronic Colloquium on Computational Complexity (ECCC) 8(18): (2001))*
If G is a (n, d, α)-expander graph, then G^k is a (n, d^k, α^k)-expander graph.

Proof. G^k is obviously d^k regular, and the eigenvalues of G^k are the eigenvalues of G to the power of k.

Proof of Fact 1. Take $\alpha > 0$ and let p be the smallest integer such that $(5\sqrt{2}/8)^p \leqslant \alpha$. Graph G^p is as required. The proof of Fact 1 is complete.

Let G be a graph on n vertices and H be a (n, d, α)-expander graph. Let t be a positive integer. We build the graph G'_t on $N = nd^{t-1}$ vertices: each vertex corresponds to a $(t-1)$-random walk $x = (x_1, \cdots, x_t)$ on H (meaning that x_1 is chosen at random, and x_{i+1} is chosen randomly in the set of neighbors of x_i), and two vertices $x = (x_1, \cdots, x_t)$ and $y = (y_1, \cdots, y_t)$ in G'_t are adjacent iff $\{x_1, \cdots, x_t, y_1, \cdots, y_t\}$ is a clique in G.

Theorem 9. *(Claims 3.15 and 3.16 in S. Hoory, N. Linial and A. Widgerson, Expander graphs and their applications, Bulletin of the AMS, 43 (4), 439–561, 2006)*
Let G be a graph on n vertices and H be a (n, d, α)-expander graph. If $b > 6\alpha$, then:

- *If $\omega(G) \leqslant bn$ then $\omega(G'_t) \leqslant (b + 2\alpha)^t N$;*
- *If $\omega(G) \geqslant bn$ then $\omega(G'_t) \geqslant (b - 2\alpha)^t N$.*

We are now able to prove the gap amplification with linear size amplification claimed in Theorem 2.

Proof of Theorem 2. Let $k = \lceil\sqrt{n}\rceil$. We modify G by adding $k^2 - n$ dummy (isolated) vertices. Let G' be the new graph. It has $n' = k^2$ vertices. Note that $n' \leqslant (\sqrt{n}+1)^2 = n + 2\sqrt{n} + 1 = n + o(n)$. Let n be such that $1 - \epsilon \leqslant n/n' \leqslant 1$ for a small ϵ. Thanks to Fact 1, we consider a (k^2, d, α)-expander graph H for a sufficiently small α (the value of which will be fixed later). According to Theorem 9 (applied on G') we build in polynomial time a graph G'_t on $N = n'd^t$ vertices such that (choosing $\alpha < b/6$):

- If $\omega(G) \leqslant bn$ then $\omega(G') = \omega(G) \leqslant bn'$, hence $\omega(G'_t) \leqslant (b + 2\alpha)^t N$;
- If $\omega(G) \geqslant an$ then $\omega(G') = \omega(G) \leqslant an'(1-\epsilon)$, hence $\omega(G'_t) \geqslant (a(1-\epsilon) - 2\alpha)^t N$.

We choose ϵ and α such that $a(1-\epsilon) - 2\alpha > b + 2\alpha$, and then t such that $(a(1-\epsilon)-2\alpha)^t/(b+2\alpha)^t \leqslant r$. The number of vertices of G'_t is clearly linear in n (first point of the theorem). Then, $b_r = (b+2\alpha)^t$ and $a_r = (a(1-\epsilon)-2\alpha)^t$ fulfil items 2, 3 and 4.

B Deferred proofs

Proof of Lemma 1. Suppose that 3Sat $\in \mathrm{PCP}_{1,\beta}[\log|\phi| + D, E]$, where $\beta \in (0,1)$, $|\phi|$ is the sum of the lengths of clauses in the 3Sat instance, D and E are constants.

Given an $\epsilon > 0$, let ϵ' such that $0 < \epsilon' < \epsilon$. Given an instance ϕ of 3 sat on n variables, we apply the sparsification lemma (with ϵ') to get $2^{\epsilon' n}$ instances ϕ_i on at most n variables. Since each variable appears at most $c_{\epsilon'}$ times in ϕ_i, the global size of ϕ_i is $|\phi_i| \leqslant c_{\epsilon'} n$.

Then for each formula ϕ_i we use the previous PCP assumption. The size of the proof is at most $E2^{|R|} = c'|\phi_i| \leqslant cn$ for some constants c', c that depend on ϵ' (where $|R| = \log n + D$ is the number of random bits) since $E2^{|R|}$ is the total number of bits that we read in the proof. Take one variable for each bit in the proof: $x_1, \cdots, x_{cn}$. For each random string R: take all the 2^E possibilities for the E variables read, and write a CNF formula which is satisfied if and only if the verifier accepts. This can be done with a formula with a constant number of clauses, say C_1, each clause having a constant number of variables, say C_2 (C_1 and C_2 depends on E).

If we consider the CNF formed by all theses CNF for all the random clauses, we get a CNF with $C_1 2^{|R|}$ clauses on variables $x_1, \cdots, x_{cn}$. The clauses are on C_2 variables but by adding a constant number of variables we can replace a clause on C_2 variables by an equivalent set of clauses on 3 variables. This way we get a 3-CNF formula and multiply the number of variables and the number of clauses by a constant, so they are still linear in n. For each R you have a set of say C'_1 clauses.

Suppose that we start from a satisfiable formula ϕ_i. Then there exists a proof for which the verifier always accepts. By taking the corresponding values for the variables x_i, and extending it properly to the new variables y, all the clauses are satisfied.

Suppose that we start from a non satisfiable formula ϕ_i. Then for any proof (i.e., any truth values of variables), the verifier rejects for a proportion of at least $(1-\beta)$ of the random strings. If the verifier rejects for a random string R, then in the set of clauses corresponding to this variable at least one clause is not satisfied. It means that among the $C_1' 2^{|R|}$ clauses (total number of clauses), at least $(1-\beta)\cdot 2^{|R|}$ are not satisfied, i.e., a fraction $(1-\beta)/C_1'$ of the clauses.

Then either $m = C_1' 2^{|R|} = O(n)$ clauses are satisfiable, or at least $m(1-\beta)/C_1'$ clauses are not satisfied by each assignment. Distinguishing between these sets in time $2^{o(m)}$ would determine whether ϕ_i is satisfiable or not in $2^{o(n)}$. Doing this for each ϕ_i would solve 3SAT in time $p(n)2^{\epsilon' n} + 2^{\epsilon' n}O(2^{o(n)}) = O(2^{\epsilon n})$ (where p is a polynomial). This is valid for any $\epsilon > 0$ so it would contradicting ETH.

Proof of Theorem 3. In the proof we will denote by N the number of vertices in a graph (to avoid confusion with the number of variables in a formula). We will show that the existence of such an algorithm for any $r' < 1$ would contradict the hardness result for MAX-3SAT in Lemma 1, hence ETH or LPC. Consider a constant $r < 1$. Let $0 < \epsilon < 1 - r$. We show that the existence of an $(r+\epsilon)$-approximation algorithm for INDEPENDENT SET running in time $f(k)N^{o(k)}$ would allow to distinguishing in time $2^{o(m)}$ between instances of MAX-3SAT where $(1-\epsilon')m$ clauses are satisfiable and instances where at most $(r+\epsilon')m$ clauses are satisfiable, for some $\epsilon' > 0$. W.l.o.g., we can assume that f is increasing, and that $f(k) \geqslant 2^k$.

Take an instance I of MAX-3SAT, let K be an integer that will be fixed later. We build a graph G_I as follows: Partition the m clauses into K groups $H_1, \cdots, H_K$ each of them containing, roughly, m/K clauses each. Each group H_i involves a number $s_i \leqslant 3m/K$ of variables. For all possible values of these variables, add a vertex in the graph G_I if these values satisfy at least $\lambda m/K$ clauses in H_i (the value of λ will also be fixed later). Finally, add an edge between two vertices if they have one contradicting variable. In particular the vertices corresponding to the same group of clauses form a clique. It is easy to see that the so-constructed graph contains $N \leqslant K 2^{3m/K}$ vertices.

The following easy claim holds.

Claim. If a variable assignment satisfies at least $\lambda m/K$ clauses in at most s groups, then it satisfies at most $\lambda m + s(1-\lambda)m/K$ clauses.

Proof of claim. An assignment as the in the claim's statement satisfies at most m/K clauses in at most s groups, and at most $\lambda m/K$ in the other $K-s$ groups, so in total at most $sm/K + (K-s)\lambda m/K = \lambda m + s(1-\lambda)m/K$, that completes the proof of the claim. ◇

Now, let us go back to the proof of the theorem. Assume an independent set of size at least t in G_I. Then one can achieve a partial solution that satisfies at least $\lambda m/K$ clauses in at least t groups. So, at least $t\lambda m/K$ clauses are satisfiable. In other words, if at most $(r+\epsilon')m$ clauses are satisfiable, then a maximum independent set in G_I has size at most $K\frac{r+\epsilon'}{\lambda}$. Suppose that at least

$(1-\epsilon')m$ clauses are satisfiable. Then, using the claim, there exists a solution satisfying at least $\lambda m/K$ clauses in at least $\frac{1-\epsilon'-\lambda}{1-\lambda}K$ groups; otherwise, it should be $\lambda m + s(1-\lambda)m/K < (1-\epsilon')m$. Then, there exists an independent set of size $\frac{1-\epsilon'-\lambda}{1-\lambda}K$ in G_I.

Now, set $K = \lceil f^{-1}(m)/(1-\epsilon^2)\rceil$. Set also $\lambda = 1-\epsilon$, and $\epsilon' = \epsilon^3$. Run the assumed $(r+\epsilon)$-approximation parameterized algorithm for Independent Set in G_I with parameter $k = (1-\epsilon^2)K$. Then, if at least $(1-\epsilon')m$ clauses are satisfiable, there exists an independent set of size at least $\frac{1-\epsilon'-\lambda}{1-\lambda}K = (1-\epsilon^3/\epsilon)K = (1-\epsilon^2)K = k$; so, the algorithm must output an independent set of size at least $(r+\epsilon)k$. Otherwise, if at most $(r+\epsilon')m$ clauses are satisfiable, the size of an independent set is at most $K\frac{r+\epsilon'}{\lambda} = K\frac{r+\epsilon^3}{1-\epsilon} = k\frac{r+\epsilon^3}{(1-\epsilon)(1-\epsilon^2)} = k(r + r\epsilon + o(\epsilon))$.

So, for ϵ sufficiently small, the algorithm allows to distinguish between the two cases of Max-3Sat (for ϵ'), i.e., whether at least $(1-\epsilon')m$ clauses are satisfiable, or at most $(r+\epsilon)m$ clauses.

The running time of the yielded algorithm is $f(k)N^{o(k)}$, but $f(k) = f((1-\epsilon^2)K) = m$, and $N^{o(k)} = N^{k/\psi(k)}$ for some increasing and unbounded function ψ, and $N^{o(k)} = (K2^{3m/K})^{k/\psi(k)} = 2^{o(m)}$.

Proof of Lemma 4. Suppose that G has an independent set S of size α. Then, S has one vertex in α sets C_i, and no vertex in the other $K-\alpha$ sets. We build a dominating set T in G' as follows: for each vertex in S we take its copy in G'. For each clique C_i without vertices in S, we take t_i and one (anyone) vertex in C'_i. The dominating set T has size $\alpha + 2(K-\alpha) = 2K-\alpha$. For each C'_i, one of its vertices in in T; so, vertices in C'_i, t_i and vertices in S_i are dominated. Now take a vertex in W_e with $e = (u,v)$, $u \in C_i$ and $v \in C_j$. If $C_i \cap S = \emptyset$ (or $C_j \cap S = \emptyset$), then $t_i \in T$ (or $t_j \in T$) and, by construction, t_i is adjacent to all vertices in W_e. Otherwise, there exist $w \in S \cap C_i$ and $x \in S \cap C_j$. Since S is an independent set, either $w \neq u$ or $x \neq v$. If $w \neq u$, by construction w (its copy in C'_i) is adjacent to all vertices in W_e and, similarly, for x if $x \neq v$. So, T is a dominating set.

Conversely, suppose that T is a dominating set of size $2K-\alpha$. Since S_i is an independent set of size $3K$, we can assume that $T \cap S_i = \emptyset$ and the same occurs with W_e. In particular, there exists at least one vertex in T in each C_i. Now, suppose that T has two different vertices u and v in the same C_i. Then we can replace v by t_i getting a dominating set (vertices in S_i are still dominated by u, and any vertex in some W_e which is adjacent to v is adjacent to t_i). So, we can assume that T has the following form: exactly one vertex in each C_i, and $K-\alpha$ vertices t_i. Hence, there are α cliques C'_i, where t_i is not in T. We consider in G the set S constituted by the α vertices in T in these α sets. Take two vertices u and v in S with, say, $u \in C'_i$ and $v \in C'_j$ (with $t_i \notin T$ and $t_j \notin T$). If there were an edge $e = (u,v)$ in G, neither u nor v would have dominated a vertex in W_e (by construction). Since neither t_i nor t_j is in T, this set would not have been a dominating set, a contradiction. So S is an independent set.

Proof of Theorem 4. In the proof of Theorem 3, we produce a graph G_I which is made of K cliques and such that: if at least $(1-\epsilon)m$ clauses are satisfiable in I, then there exists an independent set of size $(1-O(\epsilon))K$; otherwise (at most $(r+\epsilon)m$ clauses are satisfiable in I), the maximum independent set has size at most $(r+O(\epsilon))K$. The previous reduction transforms G_I in a graph G'_I such that, applying Lemma 4, in the first case there exists a dominating set of size at most $2K-(1-O(\epsilon))K = K(1+O(\epsilon))$ while, in the second case, the size of a dominating set is at least $2K-(r+O(\epsilon))K = K(2-r-O(\epsilon))$. Thus, we get a gap with parameter $k' = K(1+O(\epsilon))$. Note that the number of vertices in G'_I is $n' = n+K+3K+3K|E_I| = O(n^3)$ (where E_I is the set of edges in G_I). If we were able to distinguish between these two sets of instances in time $f(k')n'^{o(k')}$, this would allow to distinguish the corresponding independent set instances in time $f(k')n'^{o(k')} = g(k)n^{o(k)}$ since $k' = K(1+O(\epsilon)) = k(1+O(\epsilon))$ ($k = K(1-\epsilon^3)$ being the parameter chosen for the graph G_I).

Proof of Theorem 5 Again, to avoid confusion we denote in this proof by N the number of vertices in a graph. Given an $\epsilon > 0$, let ϵ' such that $0 < \epsilon' < \epsilon$. Given an instance ϕ of 3 SAT on n variables, we first apply the sparsification lemma (with ϵ') to get $2^{\epsilon' n}$ instances ϕ_i on at most n variables. Since each variable appears at most $c_{\epsilon'}$ times in ϕ_i, the global size of ϕ_i is $|\phi_i| \leqslant c_{\epsilon'} n$.

Consider a particular ϕ_i, $r > 0$ and $\delta > 0$. We use the fact that 3SAT$\in$ $\mathrm{PCP}_{1,r}[(1+o(1))\log|\phi| + D_r, E_r]$ (where D_r and E_r are constants that depend only on r), in order to build the following graph G_{ϕ_i} (see also [1]). For any random string R, and any possible value of the E_r bits read by V, add a vertex in the graph if V accepts. If two vertices are such that they have at least one contradicting bit (they read the same bit which is 1 for one of them and 0 for the other one), add an edge between them. In particular, the set of vertices corresponding to the same random string is a clique.

Assume that ϕ_i is satisfiable. Then there exists a proof for which the verifier accepts for any random string R. Take for each random string R the vertex in G_{ϕ_i} corresponding to this proof. There is no conflict (no edge) between any of these $2^{|R|}$ vertices, hence $\alpha(G_{\phi_i}) = 2^{|R|}$ (where, in a graph G, $\alpha(G)$ denotes the size of a maximum independent set).

If ϕ_i is not satisfiable, then $\alpha(G_{\phi_i}) \leqslant r2^{|R|}$. Indeed, suppose that there is an independent set of size $\alpha > r2^{|R|}$. This independent set corresponds to a set of bits with no conflict, defining part of a proof that we can arbitrarily extend to a proof Π. The independent set has α vertices corresponding to α random strings (for which V accepts), meaning that the probability of acceptance for this proof Π is at least $\alpha/2^{|R|} > r$, a contradiction with the property of the verifier.

Furthermore, G_{ϕ_i} has $N \leqslant 2^{|R|}2^{E_r} \leqslant C'|\phi_i|^{1+o(1)} = Cn^{1+o(1)}$ vertices (for some constants C, C' that depend on ϵ') since $|\phi_i| \leqslant c_{\epsilon'} n$. Then, one can see that, for any $r' > r$, an r'-approximation algorithm for INDEPENDENT SET running in time $O(2^{N^{1-\delta}})$ would allow to decide whether ϕ_i is satisfiable or not in time $O(2^{n^{1-\delta'}})$ for some $\delta' < \delta$. Doing this for each of the formula ϕ_i would allow to decide whether ϕ is satisfiable or not in time $p(n)2^{\epsilon' n} + 2^{\epsilon' n}O(2^{n^{1-\delta'}}) = O(2^{\epsilon n})$ (where p is a polynomial). This is valid for any $\epsilon > 0$ so it would contradicting ETH.

Combining this reduction with the gap amplification of Theorem 2 allows to create a gap with any constant in $(0,1)$. Since the reduction in this amplification is linear with respect to the number of vertices, we get the claimed result.

Proof of Proposition 1. Consider the following reduction from Independent Set to Bipartite Subgraph (H. U. Simon, On approximate solutions for combinatorial optimization problems, *SIAM J. Disc. Math.*, 3(2):294–310, 1990). Let $G(V,E)$ be an instance of Independent Set of order n. Construct a graph $G'(V',E')$ for Bipartite Subgraph by taking two distinct copies of G (denote them by G_1 and G_2, respectively) and adding the following edges: a vertex v_{i_1} of copy G_1 is linked with a vertex v_{j_2} of G_2, if and only if either $i=j$ or $(v_i,v_j)\in E$. G' has $2n$ vertices. Let now S be an independent set of G. Then, obviously, taking the two copies of S in G_1 and G_2 induces a bipartite graph of size $2|S|$. Conversely, consider an induced bipartite graph in G' of size t, and take the largest among the two color classes. By construction it corresponds to an independent set in G, whose size is at least $t/2$ (note that it cannot contain 2 copies of the same vertex). So, any r-approximate solution for Bipartite Subgraph in G' can be transformed into an r-approximate solution for Independent Set in G. Observe finally that the size of G' is two times the size of G.

Proof of Proposition 2. In [20] the following reduction is built. Given a graph G whose vertex set is partitioned into K cliques each of size S, and given a prime number $q>S$, a graph H_q having the following properties can be built in polynomial time: (i) the vertex set of H_q is partitioned into q^2K cliques, each of size q^3; (ii) $\alpha(H_q)\leqslant\max\{q^2\alpha(G);q^2(\alpha(G)-1)+K;qK\}$; (iii) if $\alpha(G)=K$, then $\chi(H_q)=q^3$.

Fix a ratio $r>1$, and let $r_{IS}>0$ be such that $r_{IS}+r_{IS}^2\leqslant 1/r$. Start from the graph G_{ϕ_i} produced in the proof of Theorem 5 for ratio r_{IS}. The vertex set of G_{ϕ_i} is partitioned into $K=2^{|R|}$ cliques, each of size at most 2^{E_r}. By adding dummy vertices (a linear number, since E_r is a fixed constant), we can assume that each clique has the same size $S=2^{E_r}$, so the number of vertices in G_{ϕ_i} is $N=KS=2^{|R|}2^{E_r}$.

Let $q>\max\{S,1/r_{IS}\}$ be a prime number, and consider the graph H_q produced from G_{ϕ_i} by the reduction in [20] mentioned above. If ϕ_i is satisfiable, $\alpha(G_{\phi_i})=K$ and then by the third property of the graph H_q, $\chi(H_q)=q^3$. Otherwise, by the second property $\alpha(H_q)\leqslant\max\{q^2\alpha(G_\phi);q^2(\alpha(G_\phi)-1)+K;qK\}$. Formula ϕ_i being not satisfiable, $\alpha(G_{\phi_i})\leqslant r_{IS}K$. By the choice of q, $qK\leqslant q^2r_{IS}K$, so $\alpha(H_q)\leqslant q^2r_{IS}K+K=(q^2r_{IS}+1)K$. Since the number of vertices in H_q is Kq^5, we get that $\chi(H_q)\geqslant q^5/(q^2r_{IS}+1)$. The gap created for the chromatic number in the two cases is then at least:

$$\frac{q^5}{(q^2r_{IS}+1)q^3}=\frac{1}{r_{IS}+1/q^2}\geqslant\frac{1}{r_{IS}+r_{IS}^2}\geqslant r$$

The result follows since H_q has Kq^5 vertices and q is a constant (that depends only on the ratio r and on the constant number of bits p read by V), so the size of H_q is linear in the size of G_{ϕ_i}.

Proof of Proposition 3. We combine the following theorem with a well known reduction.

Theorem 10. *[23] Under* ETH*, for every $\epsilon > 0$, and $\delta > 0$, it is impossible to distinguish between instances of* MAX 3-LIN *with m equations where at least $(1-\epsilon)m$ are satisfiable from instances where at most $(1/2+\epsilon)m$ are satisfiable, in time $O(2^{m^{1-\delta}})$.*

Consider an instance I of MAX 3-LIN on m equations. Build the following graph G_I:

- for any equation and any of the eight possible values of the 3 variables in it, add a vertex in the graph if the equation is satisfied;
- if two vertices are such that they have one contradicting variable (the same variable has value 1 for one vertex and 0 for the other one), then add an edge between them.

In particular, the set of vertices corresponding to the same equation is a clique. Note that each equation is satisfied by exactly 4 values of the variables in it. Then, the number of vertices in the graph is $N = 4m$. Consider an independent set S in the graph G_I. Since there is no conflict, it corresponds to a partial assignment that can be arbitrarily completed into an assignment τ for the whole system. Each vertex in S corresponds to an equation satisfied by τ (and S has at most one vertex per equation), so τ satisfies (at least) $|S|$ equations. Reciprocally, if an assignment τ satisfies α clauses, there is obviously an independent set of size α in G_I. Hence, if $(1-\epsilon)m$ equations are satisfiable, there exists an independent set of size at least $(1-\epsilon)m$, i.e., a vertex cover of size at most $N-(1-\epsilon)m = N(3/4+\epsilon/4)$. If at most $(1/2+\epsilon)m$ equations are satisfiable, then each vertex cover has size at least $N-(1/2+\epsilon)m = N(7/8-\epsilon/4)$.

We now handle Min-Sat problem via the following reduction (see also M. V. Marathe and S. S. Ravi, On Approximation algorithms for the minimum satisfiability problem, *Inf. Process. Lett.* 58(1): 23-29, 1996). Given a graph G, build the following instance on Min-Sat. For each edge (v_i, v_j) add a variable x_{ij}. For each vertex v_i add a clause c_i. Variable x_{ij} appears positively in c_i and negatively in c_j. Then, take a vertex cover V^* of size k; for any x_{ij} fix the variable to true if $v_i \in V^*$, to false otherwise. Consider a clause c_j with $v_j \notin V^*$. If $\overline{x_{ij}}$ is in c_j then v_i is in V^* hence x_{ij} is true; if x_{ji} is in c_j then, by construction, x_{ji} is false. So c_j is not satisfied, and the assignment satisfies at most k clauses. Conversely, consider a truth assignment that satisfies k clauses $c_{i_1}, \cdots, c_{i_k}$. Consider the vertex set $V^* = \{v_{i_1}, \cdots, v_{i_k}\}$. For an edge (v_i, v_j), if x_{ij} is set to true then c_i is satisfied and v_i is in V^*, otherwise c_j is satisfied and v_j is in V^*, so V^* is a vertex cover of size k. Since the number of clauses in the reduction equals the number of vertices in the initial graph, the result is concluded.

Proof of Corollary 5. Using LPC, the same proof as in Theorem 5 creates for each ϕ_i a graph on $N = O(n)$ variables with either an independent set of size αN (if ϕ_i is satisfiable) or a maximum independent set of size at most $\alpha/2N$ (if ϕ_i is not satisfiable). Then using expander graphs, usual arguments allows to amplify this gap from $1/2$ to any constant $r > 0$ while preserving the linear size of the instance (see Theorem 2). Results for the other problems immediately follow from the same arguments as above.

Proof of Theorem 6. Obviously, APETH(Π) is implied by APETH(Π-B). Now, suppose that APETH(Π) holds, for some ratio r. We show that APETH(Π-B) holds for the same ratio. Let $\epsilon > 0$, $\epsilon' = \epsilon/2$, and suppose that Π-B is r-approximable in time $O^*(2^{\epsilon' n})$. Then given an instance I of Π, compute $f(I, \epsilon')$ (in time $O^*(2^{\epsilon' n})$). For each of the t instances I_i, compute an r-approximate solution S_i in time $O^*(2^{\epsilon' n_i}) = O^*(2^{\epsilon' n})$, and use g to transform S_i into a solution S for I. Let S^* be the best of these solutions. We obtain S^* in time $O^*(2^{\epsilon' n} 2^{\epsilon' n}) = O^*(2^{\epsilon n})$. By item 3 of Definition 1, S^* is an r-approximation of I. We can do this for any ϵ, leading to a contradiction.

Proof of Proposition 5. Let $\epsilon > 0$, and consider the following branching algorithm, where $B' \geqslant 4$ will be precised later (as a function of ϵ):

1. remove all edges between two vertices in V_1, as well as all edges between two vertices in V_3;
2. if there exists a vertex $v \in V_1$ of degree at least B', branch on it;
3. otherwise, if there exists a vertex $v \in V_2$ of degree at least B'^2, branch on it;
4. otherwise, if there exists a vertex $v \in V_3$ of degree at least B'^3, branch on a neighbor of v.

Note that branching on a vertex v in V_1 or V_2 means that if v is taken, then v is removed from the graph, its neighbors in V_2 are transferred to V_1 (they are already dominated), while its neighbors in V_3 are removed from the graph. If v is not taken, if it is in V_1 then it is removed from the graph, and if it is in V_2 then it is transferred to V_3 (we still need to dominate it).

By principle, in a leaf of the tree, each vertex in V_1 has degree at most B', while each vertex in V_2 has degree at most B'^2, and each vertex of V_3 has degree at most B'^3. Then the graph has bounded maximum degree $B = B'^3$.

However, when branching it might be the case that only at most one vertex is removed from the graph in each branch. To show that the number of leaves in the tree is indeed sufficiently small, we change the branching measure by introducing appropriate weights on the vertices of the graph. Let $w_1 = \min\{\frac{1}{2}, \frac{1}{4} + \frac{d(v)}{4B'}\}$ be the weights of vertices in V_1, $w_2 = \min\{1, \frac{3}{4} + \frac{d(v)}{4B'}\}$ and $w_3 = 1/2$ be the weights of vertices in V_2 and V_3 respectively. Then the global weight of G is $W(G) \leqslant n$.

Consider a branching step on a vertex $v \in V_1$ corresponding to item 2 of the algorithm: if v is taken, the weight of the instance is reduced by at least $1/2 + B'/4$ ($1/2$ for v, and at least $1/4$ for each of its neighbors). If v is not taken, then the weight is reduced by $1/2$.

In a branching step on a vertex $v \in V_2$ corresponding to item 3 of the algorithm, if v is taken, the weight of the instance is reduced by at least $1 + B'^2/B' = 1 + B'$. Indeed, there is a weight-reduction of $1/2$ for v, and of at least $1/B'$ for each of its neighbors, since we know that every vertex in V_1 has degree at most $B' - 1$. If v is not taken, the weight reduces by at least $1/4$.

In a branching step on a vertex $w \in V_1 \cup V_2$ neighbor of v corresponding to item 4, when w is taken v is removed, so the degree of at least B'^3 vertices decreases by 1. Since vertices in V_1 and V_2 have degree at most $B'-1$ and B'^2-1 respectively, the total weight is reduced by at least $B'^3/B'^2 = B'$. When w is not taken, the weight is reduced by at least $1/4$.

Then, it suffices to choose B' sufficiently large such that the branching factor of these three branchings is at most 2^ϵ.

The fact that an approximate solution on a leaf can be transferred to an approximate solution to the root is completely similar to the case of independent set.

Proof of Lemma 5. Using Proposition 5, it holds that:

$$\begin{aligned}\text{APETH}(\textsc{Dominating Set}) &\Rightarrow \text{APETH}(\textsc{Generalized Dominating Set}) \\ &\Rightarrow \text{APETH}(\textsc{Generalized Dominating Set-}B)\end{aligned}$$

Consider an instance $G = (V_1, V_2, V_3, E)$ of Generalized Dominating Set-B, and use the following reduction (adapted from [24] to this generalized version). Build a graph $G' = (V', E')$ where:

- for each vertex v in $V_2 \cup V_3$, consider a clique C_v of size $|N[v] \cap (V_1 \cup V_2)|$, where each vertex of C_v corresponds to one vertex in $N[v] \cap (V_1 \cup V_2)$ (note that cliques are disjoint; if a vertex is in the neighborhood of two such vertices, there will be two different vertices in G'). Such vertices will be informally refered to as *vertices in the cliques*;
- for each vertex v in $V_1 \cup V_2$, add a vertex v' in G', and link v' to all its homologous vertices in the cliques (there is at most one per clique); hence, if $v \in V_1 \cup V_2$ has t neighbors in $V_2 \cup V_3$, v' will be linked to t vertices. Such vertices v' will be informally refered to as *vertices not in the cliques* or *vertices outside the cliques*.

Note that the size of each clique C_v is at most B, so there is at most Bn vertices in all the cliques. There are $|V_1| \leqslant n$ vertices v', so $|V'| \leqslant (B+1)n$, the reduction has linear size (with respect to n). Each vertex in a clique has degree at most $(B-1)+1 = B$, and each vertex v' has degree at most B, so G' has degree at most B.

Let D be a generalized dominating set of G. For each vertex v in $V_2 \cup V_3$, there exists a vertex $w \in D$ dominating it. We select the corresponding vertex in G' in the clique C_v. This adds up to $|V_2 \cup V_3|$ vertices. Moreover, for each vertex v in $V_1 \cup V_2$ which is not in D, we select the corresponding vertex v'; hence, we select $|V_1 \cup V_2| - |D|$ more vertices. By construction, this is an independent set S in G' of size $|S| = |V_1| + 2|V_2| + |V_3| - |D|$.

Conversely, take an independent set S of G'. Suppose that S contains no vertex from a clique C_u. Then we can add a vertex from C_u to S, and (possibly) remove the vertex v' which were adjacent to it. We get an independent set of at least the same size. By repeating the argument, we can assume that S takes one vertex from each clique C_u. Consider in G the set D of vertices that corresponds to vertices v' (which are not in cliques) in G' that are not in S. S is made of $|V_2|+|V_3|$ vertices in the cliques and $|V_1|+|V_2|-|D|$ vertices outside the cliques. So, we have $|D| = |V_1| + 2|V_2| + |V_3| - |S|$. Consider now a vertex v in $V_2 \cup V_3$. There is a vertex $w \in S$ in the clique C_v, so the vertex v' adjacent to this vertex w is not in S, hence its corresponding vertex is in D. Then D is a generalized dominating set.

Suppose that we have an r-approximate solution S in G': $S \geqslant r\alpha(G')$, we can build a solution D of size $|D| \leqslant |V_1|+2|V_2|+|V_3|-r\alpha(G') = r\gamma(G)+(1-r)(|V_1|+2|V_2|+|V_3|)$ where $\gamma(G)$ is the size of a constrained dominating set in G. Since vertices in V_1 and V_2 have degree at most B, we know that $\gamma(G) \geqslant \frac{|V_2|+|V_3|}{B}$. Note that each vertex in V_1 has at least one neighbor (otherwise, it can be removed from the graph), so that there are at most $|V_1| \leqslant B(|V_2| + |V_3|)$. Then $|V_1| + 2|V_2| + |V_3| \leqslant (B+2)(|V_2| + |V_3|) \leqslant B(B+2)\gamma(G)$. Putting all the above together, we get $|D| \leqslant \gamma(G)(r + (1-r)B(B+2))$.

Proof of Theorem 8. (i) $\Rightarrow$ (ii): We show it for Π =Independent Set-B, which is MaxSNP-complete. Suppose that for any r and any ϵ there is a parameterized r-approximation algorithm $\mathcal{A}$ in time $O(2^{\epsilon k})$. Given an instance G of Independent Set-B, we run $\mathcal{A}$ on the instance (G, k) for $k = 1$ up to $k = n$. Consider the largest k for which an independent set is given: it has size at least $\rho \cdot k$, while the optimum is at most k since no solution is output for $k + 1$. Since $k \leqslant n$, the overall iteration takes $n \cdot 2^{o(n)}$-time.

(ii) $\Rightarrow$ (iii): suppose that (iii) is false, and consider a MaxSNP-complete problem Π_2 which admits for every $\epsilon' > 0$ and every $r' < 1$ a parameterized r-approximation algorithm running in time $O^*(2^{\epsilon k})$. Then this is true for any MaxSNP problem, contradicting (ii).

Indeed, let Π_1 be a MaxSNP problem. There exists an L-reduction from Π_1 to Π_2, let α, β be the constants of the L-reduction. Let (I_1, k) be an instance of Π_1 and let $(I_2, \alpha \cdot k)$ be the instance of Π_2, where $I_2 := f(I_1)$ defined by the L-reduction. Let $r \in (0, 1)$ and $\epsilon > 0$, and let $\mathcal{A}$ be a parameterized r'-approximation of Π_2 which runs in time $O^*(2^{\epsilon' k})$ where $r' = 1 - (1-r)/(\alpha\beta) < 1$ and $\epsilon' = \epsilon/\alpha$. We present an algorithm which uses $\mathcal{A}$ as a subroutine and produces in time $O^*(2^{\epsilon k})$ a solution of Π_1 of size at least rk whenever $opt(I_1) \geqslant k$.

Let us suppose that $opt(I_1) \geqslant k$. We iterately run $\mathcal{A}$ over the instances $(I_2, \alpha k), (I_2, \alpha k - 1), \cdots$ by decreasing the parameter. Let $lb \geqslant \alpha k$ be the first integer for which that $\mathcal{A}$ returns a solution, let us call it sol_2, of size at least $r' lb$ upon (I_2, lb). Let $sol_1 := g(sol_2)$, where g is defined by the L-reduction. Note that if $opt(I_2) > \alpha k$ then $sol_2 \geqslant \alpha r' k$; if $opt(I_2) \leqslant \alpha k$, then $lb \geqslant opt(I_2)$ hence $sol_2 \geqslant r' opt(I_2)$.

Now, from the property of L-reduction, we have $opt(I_1) - sol_1 \leqslant \beta(opt(I_2) - sol_2)$, or equivalently $sol_1 \geqslant opt(I_1) - \beta(opt(I_2) - sol_2)$. By considering the two previous cases, and the fact that $opt(I_2) \leqslant \alpha opt(I_1)$ we easily get that whenever $opt(I_1) \geqslant k$, the iterative applications of $\mathcal{A}$ combined with the algorithm g returns a solution sol_1 of size at least $(1-\alpha\beta(1-r'))k = rk$. It is easily verified that the overall algorithms takes up $O(2^{\epsilon k} \cdot poly(|I_1|))$ steps.

(iii) $\Rightarrow$ (i): Suppose that for any r and any ϵ there is an r-approximation algorithm for Independent Set-B with running time $O(2^{\epsilon n})$. Given a graph G and an integer k, if $k \leqslant n/(B+1)$ we output an independent set of size $n/(B+1)$ (any maximal independent set). Otherwise, we compute an r-approximate solution S in time $O(2^{\epsilon' n}) = O(2^{\epsilon k})$ for $\epsilon' = \epsilon/(B+1)$. If $|S| \geqslant rk$ we output it, otherwise $ropt(G) \leqslant |S| < rk$, hence $opt(G) < k$. This contradicts (iii) for Independent Set-B.

Proof of Proposition 6. Given a parameter k and a set of constraints with at most c variables per constraint, the problem Max-c-Csp Above Average asks if there is a variable assignment that satisfies at least $\rho \cdot m + k$ constraints. Here ρ is the expected fraction of constraints satisfied by a uniform random assignment. In [19], the following theorem is proved.

Theorem 11. *([19]) For every $c \geqslant 2$,* Max-c-Csp Above Average *can be solved in $O(\gamma^k \cdot m)$ time, where γ is a constant depending only on c.*

Let Π be a problem in the class MaxSNP, defined in the standard way by $\max_S |\{x : \phi(x, G, S)\}|$. As shown in [24], for each of the (polynomially many) possible values x_i of x, consider the corresponding formula $\phi_i(G, S) = \phi(x_i, G, S)$. Since ϕ is fixed, this is a fixed size formula involving (at most) a fixed number t of variables (corresponding to the predicate S). The goal is then to find S satisfying the largest number of formulae ϕ_i. Let ρ_Π be the expected fraction of constraints satisfied by a uniform random assignment. It is easy to find deterministically an assignment satisfying as many formulae as a random one, so Π is ρ_Π-approximable in polynomial time. Note that Π can be interpreted as a Max-c-Csp parameterized by the number of satisfied constraints.

To get the claimed $(\rho_\Pi + \epsilon)$-approximation algorithm for $0 \leqslant \epsilon \leqslant 1 - \rho_\Pi$, we run the algorithm $\mathcal{A}$ given in Theorem 11 on the instance $(\{\phi_i : 1 \leqslant i \leqslant m\}, k')$ (where m is the number of formulas ϕ_i). We take k' so that it satisfies $\rho_\Pi \cdot m + k' = k(\rho_\Pi + \epsilon)$. If k formulae are satisfiable, then, clearly, $k(\rho_\Pi + \epsilon)$ folmulae are also satisfiable, so the algorithm will output an assignment satisfying at least this number of constraints (formulae). The running time is $\gamma^{k'} p(n)$. The claim holds since $k' = \epsilon k - \rho_\Pi(m - k)$ and $k \leqslant m$.