Hardness Amplification of Optimization Problems

Elazar Goldenberg
The Academic College of Tel Aviv Yaffo, Israel
elazargo@mta.ac.il

Karthik C. S.
Tel Aviv University, Israel
karthiks@mail.tau.ac.il

Abstract

In this paper, we prove a general hardness amplification scheme for optimization problems based on the technique of direct products.

We say that an optimization problem Π is direct product feasible if it is possible to efficiently aggregate any k instances of Π and form one large instance of Π such that given an optimal feasible solution to the larger instance, we can efficiently find optimal feasible solutions to all the k smaller instances. Given a direct product feasible optimization problem Π, our hardness amplification theorem may be informally stated as follows:

If there is a distribution $\mathcal{D}$ over instances of Π of size n such that
every randomized algorithm running in time $t(n)$ fails to solve
Π on $\frac{1}{\alpha(n)}$ fraction of inputs sampled from $\mathcal{D}$,
then, assuming some relationships on $\alpha(n)$ and $t(n)$,
there is a distribution $\mathcal{D}'$ over instances of Π of size $O(n \cdot \alpha(n))$ such that
every randomized algorithm running in time $\frac{t(n)}{poly(\alpha(n))}$ fails to solve
Π on $99/100$ fraction of inputs sampled from $\mathcal{D}'$.

As a consequence of the above theorem, we show hardness amplification of problems in various classes such as NP-hard problems like Max-Clique, Knapsack, and Max-SAT, problems in P such as Longest Common Subsequence, Edit Distance, Matrix Multiplication, and even problems in TFNP such as Factoring and computing Nash equilibrium.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography

Keywords and phrases hardness amplification, average case complexity, direct product, optimization problems, fine-grained complexity, TFNP

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.1

Related Version A full version of the paper is available at `http://arxiv.org/abs/1908.10248`.

Funding *Karthik C. S.*: Most of this work was done while the author was a student at Weizmann Institute of Science and was supported by Irit Dinur's ERC-CoG grant 772839. This work also received support from the Israel Science Foundation (grant number 552/16) and the Len Blavatnik and the Blavatnik Family foundation.

Acknowledgements We would like to thank Amir Abboud, Irit Dinur, and Eylon Yogev for discussions and comments.

1 Introduction

The widely believed conjecture $\mathsf{P} \neq \mathsf{NP}$ asserts that the class NP cannot be decided efficiently on the worst-case. That is, no polynomial time algorithm can decide the satisfiability of a CNF formula on *every* instance. However, the worst case hardness of NP still does not clarify its average-case hardness: how hard is to decide the satisfiability on a uniformly random instance.

Studying the average-case hardness of NP has a two-fold motivation. First, it may provide a more meaningful explanation than worst-case complexity about the intractability of NP-hard instances actually encountered in practice. In other words, if NP is hard only on the worst-case, then the theory of worst-case complexity that has been extensively developed over the last fifty years, might not be a good reflection of reality. Second, hardness on average is the cornerstone of modern cryptography as the security of any nontrivial cryptosystem requires some computational problem to be average-case hard (for some nice distribution). Additionally, showing average-case hardness for functions is a stepping stone towards proving strong derandomization results and the construction of pseudorandom generators.

The study of hardness amplification is the task of connecting the worst-case and average-case hardness. More specifically, based on a worst-case hardness (assumption) one would like to prove the average-case hardness of the problem.

1.1 Utopic Theorem of Hardness Amplification

A utopic theorem in the context of hardness amplification would assert that if a function is hard in the worst-case then it implies the average-case hardness for the same function against algorithms with essentially the same running time complexity. More formally it would look as follows:

▶ **Utopic Theorem 1** (A Utopic Hardness Amplification Theorem). *Let $\{f_n\}_{n\in\mathbb{N}}$ be a family of functions. Assume that every algorithm running in time $t(n)$, fails to compute f_n on at least $\gamma(n)$ fraction of inputs. Then there exists a family $\{g_n\}_{n\in\mathbb{N}}$ of functions, such that every algorithm running in time $t'(n)$, fails to compute g_n on at least $\gamma'(n)$ fraction of inputs.*

Ideally, we would like to achieve the above amplification for the following parameters[1]

1. $\gamma(n) = O(1/2^n)$ *and* $\gamma'(n) = 1/2 - O(1/2^n)$,
2. $t'(n) \approx t(n)$,
3. $\{f_n\}_{n\in\mathbb{N}} = \{g_n\}_{n\in\mathbb{N}}$.

We briefly elaborate here why we would like the above three setting of parameters in our utopic hardness amplification theorem. Item 1 would yield a worst-case to average-case reduction, and therefore extend all the lower bounds and hardness results that have been achieved in the theory of worst-case complexity for f to the average-case complexity of g. In fact, achieving $\gamma'(n) = 1/2 - O(1/2^n)$ would imply that no algorithm running in time $t'(n)$ can do much better than randomly guessing the output. Item 2 would imply that our worst-case complexity lower bounds meaningfully translate to lower bounds in the average-case. Item 3 expresses the notion of self-reducibility: if we are interested in understanding the average complexity of a problem, our hardness amplification theorem should enable us to do so by analyzing the worst-case complexity of the *same* problem. In summary, obtaining a hardness amplification result satisfying the three items is in a sense an attempt to bridge the gap between theory and practice. Finally, we remark that our utopic theorems would gain more importance if the family of functions for which we show hardness amplification are natural (in some broad sense).

Specifically, if we prove such a theorem for the family of deciding satisfiability of CNF formulas, then we get that the assumption that P $\neq$ NP implies that every polynomial time algorithm fails to decide satisfiability on slightly more that half of the CNF formulas –

[1] In order to succinctly specify the desirable parameters of a hardness amplification theorem, we assume here that f_n and g_n are Boolean functions.

a highly non-trivial and very desirable result that would pave the way for the construction of one-way functions from (weak) worst-case assumptions. However, as we wake up from the dream of a utopia, one may wonder if such a result can even be achieved [7].

Remarkably, nearly three decades ago, Lipton [26, 9] proved the above type of (utopic) theorem for the function of computing the permanent (a #P-complete problem) against probabilistic polynomial time algorithms. Trevisan and Vadhan [32] following a line of works [3, 24, 29] proved such an amplification result for PSPACE and almost proved such an amplification result for the class EXP (they couldn't achieve Item 3). For the class NP we are far from proving a strong hardness amplification result, and there are some known barriers while trying to convert worst-case NP-hardness into average-case hardness (see e.g. [6]). More recently, strong hardness amplification results have been proved for functions in P [4, 16, 17]. We also note that hardness amplification results have also been shown for one-way functions [34, 15, 5].

Given the above state-of-the-art picture, we raise a few natural questions and address them in this paper. There are many problems that are hard in the worst-case but easy on average. For example, 3-coloring is a well-known NP-hard problem, but it is an easy exercise to show that it can be solved in linear time with high probability on a random graph. This motivates us to distinguish within worst-case hard problems as to which of them remain hard on average. One way to go about this task is to identify which worst-case hard problem admits a hardness amplification theorem.

For which problems can we amplify hardness?
Can we identify a mathematical structure that allows us to amplify hardness?

The latter question has been implicitly addressed in literature (for example, if the problem has algebraic structure like in the case of computing permanent [26] or counting k-cliques [17]), but are quite specific and not broad enough to capture the class of problems that we believe are hard on average. In Section 1.2.1 we address the above two questions.

Next, we turn our attention to NP-hard problems. In a beautiful paper, O'Donnell [28] initiated the study of (non-uniform) hardness amplification in NP. His result was improved by [19] who showed that if for some function in NP (with n inputs) we have that any $s(n)$ size circuit fails to compute the function correctly on $1/poly(n)$ fraction of the inputs then, the hardness can be amplified to show that there is some function in NP such that any $s(\sqrt{n})^{\Omega(1)}$ size circuit fails to compute the function on $1/2 - 1/s(\sqrt{n})^{\Omega(1)}$ fraction of the inputs. However, the best uniform hardness amplification results (against algorithms as opposed to circuits) that have been achieved do not match the parameters of [19]: Trevisan [31, 8] improving on his previous work [30] showed that we can amplify hardness from $1/poly(n)$ to $1/2 - 1/polylog(n)$ for NP against randomized polynomial time algorithms (later extended to deterministic algorithms in [18]). However, it is important to note that all these hardness amplification results are for decision problems, and this leads us to our next question, do we gain anything by moving to search problems, or more precisely to the focus of this paper, to optimization problems?

Can we improve our hardness amplification results for optimization problems?
Can we prove stronger uniform hardness amplification results for MaxSAT*?*

Arguably, optimization problems are as natural as decision problems, but are strictly harder from the point of view of computational complexity. Does this mean we can either give simpler proofs of hardness amplification for optimization problems or prove stronger results? We address the above questions in Section 1.2.2.

We now shift our focus to the class P. As mentioned earlier, we have strong worst-case to average-case results established for problems in P [4, 17]. The drawback however is that they are all for counting problems. This is indeed inherent as the underlying technique these works use are the same as the one used to show worst-case to average-case reduction for the permanent problem. While counting the number of k-cliques (the problem considered in [17]) is a natural problem, and therefore hardness amplification for that problem is interesting, it still leaves the door open for proving hardness amplification for the search problem of just finding one k-clique in a graph (an easier problem and thus harder to amplify hardness).

Can we prove hardness amplification results for natural search problems in P*?*

Moreover, there is a barrier [1] to using the algebraic techniques of [4, 17] to obtain hardness amplification for important problems studied in fine-grained complexity such as computing the Longest Common Subsequence (LCS) and Edit Distance (Edit-Distance) for a pair of strings. In particular, if these string problems can be represented using low-degree polynomials, then we could obtain small speedups by using the polynomial method [11], which would imply new circuit lower bounds [2]. This suggests we might need to look beyond these algebraic techniques for proving hardness amplification for these string problems. Is there a different technique to prove hardness amplification in P? We address these aforestated questions in Section 1.2.3.

1.2 Our Results

Our main contribution is a general hardness amplification theorem for optimization problems which we state in Section 1.2.1. Next, we apply our main theorem to various problems. In Section 1.2.2 we state our hardness amplification theorems for various NP-hard problems such as Knapsack and MaxSAT. In Section 1.2.3 we state our hardness amplification theorems for various string problems in P such as LCS and Edit-Distance. Finally, in Section 1.2.4 we state our hardness amplification theorems for various problems in TFNP (believed to not be in P) such as Factoring and computing Nash equilibrium.

1.2.1 Hardness Amplification of Optimization Problems

Aggregation is a key tool in the field of hardness amplification. If a function f is hard to compute on a tiny fraction of the domain, then, intuitively, computing multiple instances of f in one shot should be hard on a larger fraction of the inputs. More formally, for a function $f : [N] \to \Sigma$ and $k \in \mathbb{N}$, its k-direct product encoding is defined as a function $f^{(k)} : [N]^k \to \Sigma^k$ mapping each tuple $(x_1, \ldots, x_k)$ into $(f(x_1), \ldots, f(x_k))$. Using standard techniques one can show a "direct product theorem" stating that if f is hard against $t(n)$ running-time algorithms on $\alpha(n)$-fraction of the domain, then $f^{(k)}$ is hard against $t'(n)$ running-time algorithms on $\approx k \cdot \alpha(n)$-fraction of its domain. But in order to utilize such a direct product result, we need to be able to stitch k-instances into a single (larger) instance. To address this task we introduce the following notion of direct product feasibility.

▶ **Definition 1** (Direct Product Feasibility; Informal statement)**.** *Let* Π *be an optimization problem. We say that* Π *is* (S, T)*-direct product feasible*[2] *if the exists a pair of deterministic algorithms* (Gen, Dec) *satisfying the following:*

[2] In the formal definition of direct product feasibility, it is defined for a pair of optimization problems (Π, Λ) for technical reasons which will be addressed later in Section 1.2.3. In the case $\Pi = \Lambda$ we formally call it as self direct product feasible and this notion coincides with the informal definition given here. For most of the applications given in this paper, self direct product feasibility notion suffices.

- Gen *takes as input* k *instances* $(I_1, \ldots, I_k)$ *of* Π *each of size* n *and outputs an instance* I' *of* Π *of size* $S(n,k)$.
- Dec *gets as input* $(I_1, \ldots, I_k)$, *the instance* I' *which is the output of* Gen *on input* $(I_1, \ldots, I_k)$, *an optimal solution for* I', *and* $i \in [k]$. *It outputs an optimal solution for the instance* I_i.
- *The running time of* Gen *and* Dec *is bounded by* $T(n,k)$.

Our main theorem is about hardness amplification for an arbitrary direct product feasible problem Π. In particular we show that if Π is hard against $t(n)$ running time randomized algorithms on a tiny fraction of the domain, then Π is hard on a much larger fraction of the domain against randomized algorithms with a similar running time.

▶ **Theorem 2** (Informal Statement). *Let* Π *be* (S,T)*-direct product feasible. Let* $\mathcal{D}(n)$ *be an efficiently samplable distribution over the instances of* Π *of size* n. *Assume the following:*

- *Any* $t(n)$ *running-time algorithm with success probability at least* $2/3$ *fails to compute an optimal solution on at least* $\alpha(n)$*-fraction of the inputs sampled from* $\mathcal{D}$.
- *Fix* $k = poly((\alpha(n))^{-1})$. *Then we have* $T(n,k) = o(t(n))$.
- *We can (deterministically) decide the optimality of a given solution to any instance in* $o(t(n))$ *time.*

Then there exists an efficiently samplable distribution $\mathcal{D}'(S(n,k))$ *over instances of* Π *of size* $S(n,k)$ *such that every* $t(n)$ *running-time algorithm with success probability at least* $2/3$ *fails to compute an optimal solution on at least* 99% *of the inputs sampled from* $\mathcal{D}'$.

Naturally, the distribution $\mathcal{D}'$ is defined as follows: Draw k independent samples $I_1, \ldots, I_k$ from $\mathcal{D}$, and output $\mathsf{Gen}(I_1, \ldots, I_k)$. The proof of our main theorem is based on a reduction using an oracle access to an algorithm that solves $\mathcal{D}'$ on 1% of the inputs, we convert it into an algorithm solving $\mathcal{D}$ on greater than $1 - \alpha(n)$ fraction of the inputs. The reduction is uniform, so in case that the algorithms $(\mathsf{Gen}, \mathsf{Dec})$ are uniform we get a uniform hardness amplification result.

Another key point is that our hardness amplification is a self-reduction, i.e., if a problem is somewhat hard against one distribution $\mathcal{D}$, then the same problem is much harder against a different distribution $\mathcal{D}'$.

To the best of our knowledge, this is the first result to study hardness amplification for optimization problems. It opens avenues to prove results in various subclasses as we will see in subsequent subsections.

1.2.2 Hardness Amplification for NP-hard Problems

In the NP world, we generalize the results of [28, 31] to optimization problems. In particular we show that if MaxSAT is hard to solve on $1/poly(n)$ fraction of the inputs of samples drawn from some samplable distribution $\mathcal{D}$. Then there exists a samplable distribution $\mathcal{D}'$ such that solving MaxSAT on $\mathcal{D}'$ is hard on at least $99/100$-fraction of the samples.

▶ **Theorem 3** (Informal Statement). *Let* $\mathcal{D}(n)$ *be a distribution over* 3-CNF *formulas with* n *variables and* $poly(n)$ *clauses, such that for every randomized algorithm* $\mathcal{A}$ *running in time* $poly(n)$, *we have:*

$$\Pr_{\Psi \sim \mathcal{D}}[\mathcal{A} \text{ finds a optimal assignment for } \Psi \text{ w.p. } \geq 2/3] \leq 1 - 1/poly(n).$$

Then there exists a distribution $\mathcal{D}'(n')$ over 3-CNF *formulas with n' variables poly(n') clauses, such that for every randomized algorithm $\mathcal{A}'$ running in time poly(n'), we have:*

$$\Pr_{\Psi' \sim \mathcal{D}'}[\mathcal{A}' \text{ finds a optimal assignment for } \Psi' \text{ w.p.} \geq 2/3] \leq 0.01.$$

Moreover, if $\mathcal{D}(n)$ is poly(n)-samplable then $\mathcal{D}'(n')$ is poly(n')-samplable.

Observe that the failure probability on $\mathcal{D}'$ is much larger than in [28, 31] and can even tend to 0 for a proper choice of our parameters. This can be achieved since we deal with optimization problems instead of decision problems.

We also remark that our reduction and the proof correctness are much simpler, and in particular we do not rely the hard core set lemma [20], a powerful and non-trivial key tool in the previous known proofs.

Our result easily extends into other NP-hard problems such as finding the largest clique in a graph, or finding smallest dominating set or vertex cover of a graph, etc.

However, there are other NP-hard problems for which establishing a hardness amplification result through Theorem 2 is not easy. A special highlight is that of proving such a result for the Knapsack problem, as it isn't immediately clear if it's direct product feasible for reasonable range of parameters. This is because, for the Knapsack problem, when we aggregate instances in the natural way, optimal solutions of one instance may interfere with other instances. Nonetheless, with some care, the direct product feasibility of Knapsack problem was established.

The Exponential Time Hypothesis (ETH) [22, 23, 10] asserts that that we cannot decide whether a given 3-CNF is satisfiable in time which is sub-exponential in the number of variables. That is a worst case assumption, and it raises a natural question arises: Can we prove stronger hardness amplification result based on ETH? In fact, can we prove a worst case to an average case hardness amplification based on ETH?

Our next theorem is a step towards proving such a worst-case to an average case reduction for MaxSAT.

▶ **Theorem 4** (Informal Statement). *Let $\mathcal{D}(n)$ be a distribution over* 3-CNF *instances with n variables and $O(n)$-clauses, such that for every randomized algorithm $\mathcal{A}$ running in time $2^{o(n)}$, we have:*

$$\Pr_{\Psi \sim \mathcal{D}}[\mathcal{A} \text{ finds an optimal assignment for } \Psi \text{ w.p.} \geq 2/3] \leq 1 - \frac{1}{2^{o(n)}}.$$

Then there exists a distribution $\mathcal{D}'(n')$ over 3-CNF *instances with n' variables and $2^{o(n')}$ clauses, such that for every polynomial time randomized algorithm $\mathcal{A}'$, we have:*

$$\Pr_{\Psi' \sim \mathcal{D}'}[\mathcal{A}' \text{ finds an optimal assignment for } \Psi' \text{ w.p.} \geq 2/3] \leq 0.01.$$

Heally et al. [19] proved a similar result for the non-uniform case. Our result is stronger in the sense that we use a weaker assumption: we rely on the ETH that is assuming that every *uniform* algorithm fails on $1/2^{o(n)}$-fraction of inputs. While Heally et al. use similar assumption against non-uniform algorithms.

1.2.3 Hardness Amplification in P

We investigate hardness amplification in P and can show results for string problems, such as LCS and Edit-Distance, which were not possible in previous works.

▶ **Theorem 5** (Informal statement). *Fix $\varepsilon > 0$. Let $\mathcal{D}(n)$ be an efficiently samplable distribution over the instances of* LCS/Edit-Distance *of length n. Assume that any $n^{2-\varepsilon}$ running-time algorithm with success probability at least $2/3$ fails to compute an optimal alignment on at least $1/n^{o(1)}$-fraction of the inputs sampled from $\mathcal{D}$. Then for some $\varepsilon' > 0$ there exists an efficiently samplable distribution $\mathcal{D}'(n^{1+o(1)})$ over instances of* LCS/Edit-Distance *of size $n^{1+o(1)}$ such that every $n^{2-\varepsilon'}$ running-time algorithm with success probability at least $2/3$ fails to compute an optimal solution on at least 99% of the inputs sampled from $\mathcal{D}'$.*

Recall from earlier in this section that Abboud [1] had pointed out a barrier to obtaining a result such as above, through algebraic techniques. Another similarity search problem that is studied along with LCS and Edit-Distance, is the problem of computing the Fréchet distance between two (discrete) curves. Strangely, this problem resists all natural approaches to show that it is direct product feasible. Therefore, it is an interesting question as to whether it is possible to show that it is direct product feasible (for relevant range of parameters) or whether it is a candidate for a problem that is not direct product feasible.

Additionally, we show hardness amplification for a very different kind of problem, that of computing the product of two matrices. We highlight this problem, as it does not directly follow from our main theorem (i.e., Theorem 2). Elaborating, a detail that was brushed under the carpet while discussing Theorem 2 was that, given an instance of an optimization problem and a candidate solution, we need to able to efficiently compute the value of the objective of the candidate solution for that instance. This naturally holds for all the problems considered in this paper except the task of computing the product of two matrices, i.e., we do not know a way to *deterministically* verify if the product of two matrices is equal to a given third matrix, which is significantly faster than actually multiplying the two given matrices and checking if it's equal to the third matrix [25, 33]. Nonetheless, we modify the proof of Theorem 2 to handle this issue.

1.2.4 Hardness Amplification in TFNP

Total problems (with not necessarily efficient verification of totality) are essentially equivalent to Optimization problems. The class TFNP is special as it is in an informal sense the intersection of Search NP and Optimization problems. Problems in TFNP capture problems in various areas such as game theory, cryptography, computational geometry, etc. We show that our general theorem can be applied to TFNP problems as well, and as an example show it for the Factoring problem and the End of a Line problem. The latter hardness amplification result directly implies the hardness amplification of various problems in game theory such as computing an approximate Nash equilibrium.

1.3 Open Problems

Our work leaves open several questions. We state a few of them below.

1.3.1 Stronger Hardness Amplification

In Theorem 4 we showed that if MaxSAT is hard to compute on $1 - 1/2^{o(n)}$-fraction of inputs for sub-exponential time algorithm, then there exists a distribution on which it is hard on a constant fraction of inputs for algorithms running in time $n^{\omega(1)}$. A natural open question is the following:

Can we improve Theorem 4 and get hardness amplified against sub-exponential time algorithms (instead of super-polynomial time algorithms)?

It seems to us that derandomized direct product theorems may serve as the key tool to address the above question (for example, see [21]). In particular, if one can prove a (strongly) derandomized version of [14] then it might be possible to both aggregate sub-exponentially many instances succinctly and sample from the (derandomized) direct product distribution efficiently.

1.3.2 Direct Product Feasibility

In this paper, we were able to show direct product feasibility for certain problems quite easily (for example, see Theorems 3 and 5), but had to work harder to prove them for some other problems (for example, Knapsack and Matrix Multiplication), and in some problem(s) were unable to establish the property of direct product feasibility (for example, computing Fréchet distance). This leads us to the following question.

Can we pinpoint what property of a problem makes it possible to establish direct product feasibility?

1.3.3 Gap Amplification versus Hardness Amplification

Direct Product theorems are key ingredients for both gap amplification and hardness amplification. Also, there are many philosophical similarities in the techniques known in literature of the aforementioned two kinds of amplifications. Thus we can ask the following (ambitious) question:

Can we obtain a trade-off between gap amplification and hardness amplification?

In particular, can we show that if one problem is hard to approximate on worst case within some factor $\alpha > 0$, then it is hard to approximate within a factor $\alpha/100$ on average? We note here that Feige [13], did answer the converse of this question, i.e., he used average case hardness assumptions to prove hardness of approximation results for various problems in NP.

It seems to us that analyzing the operation of performing a small perturbation on the given instance may be the right direction to proceed. Elaborating, consider a (worst case) hard distribution over gap instances of some problem. If we build a new distribution, which samples from the aforementioned distribution, then performs a small perturbation on the sampled gap instance, and outputs the perturbed instance, then we would still retain most of the gap in the instance sampled from the new distribution, but on the other hand, the fraction of instances on which it is hard to solve the problem should increase significantly. It would be interesting if this intuition/approach could be made to work.

A related question is to ask if we can improve our result in Theorem 4 (for example, by making progress on the question detailed in Section 1.3.1) using Gap-ETH [12, 27] (instead of ETH)?

1.3.4 Average Case Hard Problems in P

In this paper, we looked at average case hardness of some problems in P against some efficiently sampleable distribution but one can ask if we can achieve more.

Can we show for some natural problem in P *that it is hard to solve for the uniform distribution?*

Another important question stemming from cryptography [4] is whether we can construct a *fine-grained* one way function from worst case assumptions?

2 Proof Overview

We provide a proof overview for our hardness amplification result for the problem of finding the maximum clique in a graph and then in the subsequent section we will show how our general result (i.e., Theorem 2) would follow.

2.1 Hardness Amplification for Max Clique

To illustrate the main ideas behind our scheme let us focus on MaxCLIQUE, the problem of finding the largest clique in a given graph G.

Assume the existence of a distribution $\mathcal{D}$ over graphs on n vertices which is somewhat hard to compute. That is for every *randomized* algorithm $\mathcal{A}$ running in time $poly(n)$, we have

$$\Pr_{G\sim\mathcal{D}}[\mathcal{A} \text{ finds max-clique in } G \text{ w.p. } \geq 2/3] \leq 1 - 1/n. \tag{1}$$

We would like to prove the existence of a new distribution $\mathcal{D}'$ over graphs on $poly(n)$ vertices which is much harder to compute. That is, for every randomized algorithm $\mathcal{A}'$ running in time $poly(n')$, we have:

$$\Pr_{G'\sim\mathcal{D}'}[\mathcal{A}' \text{ finds max-clique in } G' \text{ w.p. } \geq 2/3] \leq 0.01. \tag{2}$$

Moreover if $\mathcal{D}$ is $poly(n)$-time samplable, then so is $\mathcal{D}'$.

Construction of New Distribution

$\mathcal{D}'$ samples a graph H as follows:

1. Independently sample $G_1, \ldots, G_k$ from $\mathcal{D}$, where $k = poly(n)$.
2. Define $V(H) = V(G_1) \dot\cup \cdots \dot\cup V(G_k)$.
3. For every $i \in [k]$, connect the vertices in $V(G_i)$ using the original edges in G_i.
4. For every $i, j \in [k]$ such that $i \neq j$, insert all the possible edges between G_i and G_j.
5. Output H.

Clearly, if $\mathcal{D}$ is $poly(n)$-time samplable, then so is $\mathcal{D}'$. Now assume for sake of contradiction, that there exists $\mathcal{A}'$ running in time $poly(n')$, violating Equation (2). We show the existence of an algorithm $\mathcal{A}$ running in time $poly(n)$ violating Equation (1).

The algorithm $\mathcal{A}$ on input graph G with n vertices is defined as follows:

1. Let $\mathcal{S}$ be an empty set.
2. Repeat following $O(n)$ times.
   a. Pick randomly $i \in [k]$.
   b. Independently sample $G_1, \ldots, G_{i-1}, G_{i+1}, \ldots G_k$ from $\mathcal{D}$.
   c. Construct H setting G_i to be G.
   d. Find clique in H using $\mathcal{A}'$.
   e. Restrict clique in H to the vertices of G and add it to $\mathcal{S}$.
3. Output the largest clique in $\mathcal{S}$.

Clearly, the running time of $\mathcal{A}$ is $poly(n)$, as $n' = poly(n)$ and the running time of $\mathcal{A}'$ is $poly(n')$. Our first observation is that for any graph H constructed by $\mathcal{A}$, and for every $i \in [k]$ the restriction of a maximal clique in H into G_i, is a maximal clique for G_i.

Let $\mathcal{A}_0$ be one iteration of step 2 of $\mathcal{A}$. If we show that $\mathcal{A}_0$ outputs maximum clique w.p. $\Omega(1/n)$ on $1 - 1/n$ fraction of samples from $\mathcal{D}$ then, $\mathcal{A}$ outputs maximum clique w.p. $2/3$ on $1 - 1/n$ fraction of samples from $\mathcal{D}$.

Now, observe that if instead of planting the given input graph G as the i-th subgraph of H, we were planting a uniformly random sample of $\mathcal{D}$, then we get a graph H which is drawn according to $\mathcal{D}'$. Consequently, if that was the case, then the success probability of $\mathcal{A}_0$ was equal the probability of $\mathcal{A}'$ and we were done.

Let $\mathcal{D}'_G$ denote the marginal distribution over H, where the graph G is planted at a random coordinate $i \in [k]$. We conclude the proof by showing that for $1 - 1/n$-fraction of instances G drawn from $\mathcal{D}$ we have:

$$\begin{aligned}&\Pr_{G' \sim \mathcal{D}'_G}\left[\mathcal{A}' \text{ finds max-clique in } G' \text{ w.p. } \geq 2/3\right] \\ \geq\ &\frac{\Pr_{G' \sim \mathcal{D}'}\left[\mathcal{A}' \text{ finds max-clique in } G' \text{ w.p. } \geq 2/3\right]}{2}.\end{aligned}$$

Towards this goal we use a result by Feige and Kilian [14] that was proven in the context of parallel repetition. Under minor manipulations their result can be stated as follows:

Let X be a universe and $\mathcal{T}$ be a distribution over X. Let $f : X^k \to \{0,1\}$. Define

$$\mu = \mathop{\mathbb{E}}_{x^k \sim \mathcal{T}^k}\left[f\left(x^k\right)\right],$$

$$\mu_x = \mathop{\mathbb{E}}_{i \in [k], x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k \sim \mathcal{T}}\left[f(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots x_k)\right].$$

$$\Pr_{x \sim \mathcal{T}}\left[|\mu_x - \mu| \geq k^{-1/6}\right] \leq k^{-1/6}, \tag{3}$$

To conclude the result, set X as the set of graphs with n vertices, and $\mathcal{T}$ be the distribution $\mathcal{D}$. We have $\mathcal{D}' = \mathcal{D}^k$. Define $f : X^k \to \{0,1\}$ by:

$$f(G') = 1 \iff \mathcal{A}' \text{ finds a maximal clique in } G \text{ w.p. } \geq 2/3.$$

In these notations,

$$\begin{aligned}\mu &= \Pr_{G' \sim \mathcal{D}'}\left[\mathcal{A}' \text{ finds max-clique in } G' \text{ w.p. } \geq 2/3\right] \\ \mu_x &= \Pr_{G' \sim \mathcal{D}'_G}\left[\mathcal{A}' \text{ finds max-clique in } G' \text{ w.p. } \geq 2/3\right].\end{aligned}$$

By an application of (3), and a proper choice of k, we get that for all but at most $k^{1/6}$-fraction of graphs G drawn according to $\mathcal{D}$, the success probability of $\mathcal{A}'$ on $\mathcal{D}'_G$ is $\Omega(n)$, as claimed.

2.2 Abstraction

In the previous subsection, we showed the main ingredients used for proving hardness amplification for the task of finding a maximal clique in a given graph. What were the properties of MaxCLIQUE that we utilized to prove the result?

One property that we used was that if we are given k input graphs $G_1, \ldots, G_k$, there exists an efficient way to construct a large graph H such that a maximal clique in H induces a maximal clique on each of the graphs G_i. The second property was that given a maximal clique in H there exists an efficient algorithm to construct a maximal clique on each of the graphs G_i.

These two properties are captured in Definition 1: The first property of a problem Π being Direct Product feasible is the existence of an efficient algorithm Gen stitching k instances $I_1, \ldots, I_k$ of Π into a larger instance I' of Π, such that: an optimal solution for I' induces an optimal solution for each of the instances I_i. The second property the existence of an efficient algorithm Dec converting an optimal solution for I' into an optimal solution of I'.

Once we show Π is Direct Product feasible then the rest of the proof goes through. Indeed, assuming the existence of a distribution $\mathcal{D}$ on instances of Π for which any efficient algorithm fails to compute on $1 - 1/n$ fraction of inputs, we define the distribution $\mathcal{D}', {\mathcal{D}'}_I$ as follows:

- $\mathcal{D}'$ is the k-product distribution of $\mathcal{D}$, where we pick k random samples from $\mathcal{D}'$ independently.
- ${\mathcal{D}'}_I$ is the distribution where we pick uniformly at random $i \in [k]$, and independently sample $I_1, \ldots, I_{i-1}, I_{i+1}, \ldots I_k$ from $\mathcal{D}$. Finally, we construct I' by setting I_i to be I.

Now we can use [14] to show that for most instances $I \sim \mathcal{D}$ to connect the success probability of $\mathcal{A}'$ on $\mathcal{D}'$ and ${\mathcal{D}'}_I$, to conclude the proof.

Remark about Direct Product results and Hardness Amplification

The direct product lemma at the heart of most hardness amplification results is the XOR lemma [34]. But here we critically use the fact the problem is total, so at the surface at least, our results are incomparable to the hardness amplification results for NP and EXP obtained via XOR lemmas.

References

1. Amir Abboud. Personal communication, 2019.
2. Amir Abboud, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Ryan Williams. Simulating branching programs with edit distance and friends: or: a polylog shaved is a lower bound made. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 375–388, 2016. doi:10.1145/2897518.2897653.
3. László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3:307–318, 1993. doi:10.1007/BF01275486.
4. Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Average-case Fine-grained Hardness. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 483–496, New York, NY, USA, 2017. ACM. doi:10.1145/3055399.3055466.
5. Andrej Bogdanov and Alon Rosen. Input Locality and Hardness Amplification. *J. Cryptology*, 26(1):144–171, 2013. doi:10.1007/s00145-011-9117-y.
6. Andrej Bogdanov and Luca Trevisan. Average-Case Complexity. *Foundations and Trends in Theoretical Computer Science*, 2(1), 2006. doi:10.1561/0400000004.
7. Andrej Bogdanov and Luca Trevisan. On Worst-Case to Average-Case Reductions for NP Problems. *SIAM J. Comput.*, 36(4):1119–1159, 2006. doi:10.1137/S0097539705446974.
8. Joshua Buresh-Oppenheim, Valentine Kabanets, and Rahul Santhanam. Uniform Hardness Amplification in NP via Monotone Codes. *Electronic Colloquium on Computational Complexity (ECCC)*, 13(154), 2006. URL: http://eccc.hpi-web.de/eccc-reports/2006/TR06-154/index.html.
9. Jin-yi Cai, Aduri Pavan, and D. Sivakumar. On the Hardness of Permanent. In *STACS 99, 16th Annual Symposium on Theoretical Aspects of Computer Science, Trier, Germany, March 4-6, 1999, Proceedings*, pages 90–99, 1999. doi:10.1007/3-540-49116-3_8.
10. Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. A Duality between Clause Width and Clause Density for SAT. In *21st Annual IEEE Conference on Computational Complexity (CCC 2006), 16-20 July 2006, Prague, Czech Republic*, pages 252–260, 2006. doi:10.1109/CCC.2006.6.

11 Timothy M. Chan and Ryan Williams. Deterministic APSP, Orthogonal Vectors, and More: Quickly Derandomizing Razborov-Smolensky. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1246–1255, 2016. doi:10.1137/1.9781611974331.ch87.

12 Irit Dinur. Mildly exponential reduction from gap 3SAT to polynomial-gap label-cover. *ECCC*, 23:128, 2016. URL: http://eccc.hpi-web.de/report/2016/128.

13 Uriel Feige. Relations between average case complexity and approximation complexity. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 534–543, 2002. doi:10.1145/509907.509985.

14 Uriel Feige and Joe Kilian. Two-Prover Protocols - Low Error at Affordable Rates. *SIAM J. Comput.*, 30(1):324–346, 2000. doi:10.1137/S0097539797325375.

15 Oded Goldreich, Russell Impagliazzo, Leonid A. Levin, Ramarathnam Venkatesan, and David Zuckerman. Security Preserving Amplification of Hardness. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I*, pages 318–326, 1990. doi:10.1109/FSCS.1990.89550.

16 Oded Goldreich and Guy N. Rothblum. Worst-case to Average-case reductions for subclasses of P. *Electronic Colloquium on Computational Complexity (ECCC)*, 24:130, 2017. URL: https://eccc.weizmann.ac.il/report/2017/130.

17 Oded Goldreich and Guy N. Rothblum. Counting t-Cliques: Worst-Case to Average-Case Reductions and Direct Interactive Proof Systems. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 77–88, 2018. doi:10.1109/FOCS.2018.00017.

18 Parikshit Gopalan and Venkatesan Guruswami. Hardness amplification within NP against deterministic algorithms. *J. Comput. Syst. Sci.*, 77(1):107–121, 2011. doi:10.1016/j.jcss.2010.06.008.

19 Alexander Healy, Salil P. Vadhan, and Emanuele Viola. Using Nondeterminism to Amplify Hardness. *SIAM J. Comput.*, 35(4):903–931, 2006. doi:10.1137/S0097539705447281.

20 Russell Impagliazzo. Hard-Core Distributions for Somewhat Hard Problems. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*, pages 538–545, 1995. doi:10.1109/SFCS.1995.492584.

21 Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. New Direct-Product Testers and 2-Query PCPs. *SIAM J. Comput.*, 41(6):1722–1768, 2012. doi:10.1137/09077299X.

22 Russell Impagliazzo and Ramamohan Paturi. On the Complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. Preliminary version in CCC'99. doi:10.1006/jcss.2000.1727.

23 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which Problems Have Strongly Exponential Complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. Preliminary version in FOCS'98. doi:10.1006/jcss.2001.1774.

24 Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E Requires Exponential Circuits: Derandomizing the XOR Lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 220–229, 1997. doi:10.1145/258533.258590.

25 Marvin Künnemann. On Nondeterministic Derandomization of Freivalds' Algorithm: Consequences, Avenues and Algorithmic Progress. In *26th Annual European Symposium on Algorithms, ESA 2018, August 20-22, 2018, Helsinki, Finland*, pages 56:1–56:16, 2018. doi:10.4230/LIPIcs.ESA.2018.56.

26 Richard J. Lipton. New Directions In Testing. In *Distributed Computing And Cryptography, Proceedings of a DIMACS Workshop, Princeton, New Jersey, USA, October 4-6, 1989*, pages 191–202, 1989. doi:10.1090/dimacs/002/13.

27 Pasin Manurangsi and Prasad Raghavendra. A Birthday Repetition Theorem and Complexity of Approximating Dense CSPs. *CoRR*, abs/1607.02986, 2016. arXiv:1607.02986.

28 Ryan O'Donnell. Hardness amplification within NP. *J. Comput. Syst. Sci.*, 69(1):68–94, 2004. doi:10.1016/j.jcss.2004.01.001.

29 Madhu Sudan, Luca Trevisan, and Salil P. Vadhan. Pseudorandom Generators without the XOR Lemma. *J. Comput. Syst. Sci.*, 62(2):236–266, 2001. doi:10.1006/jcss.2000.1730.

30 Luca Trevisan. List-Decoding Using The XOR Lemma. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 126–135, 2003. doi:10.1109/SFCS.2003.1238187.

31 Luca Trevisan. On uniform amplification of hardness in NP. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 31–38, 2005. doi:10.1145/1060590.1060595.

32 Luca Trevisan and Salil P. Vadhan. Pseudorandomness and Average-Case Complexity Via Uniform Reductions. *Computational Complexity*, 16(4):331–364, 2007. doi:10.1007/s00037-007-0233-x.

33 Virginia Vassilevska Williams and R. Ryan Williams. Subcubic Equivalences Between Path, Matrix, and Triangle Problems. *J. ACM*, 65(5):27:1–27:38, 2018. doi:10.1145/3186893.

34 Andrew Chi-Chih Yao. Theory and Applications of Trapdoor Functions (Extended Abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 80–91, 1982. doi:10.1109/SFCS.1982.45.