CS 354: Unfulfilled Algorithmic Fantasies May 22, 2019

Lecture 16: Hardness of Densest-k-Subgraph (DkS)

Lecturer: Aviad Rubinstein *Scribe: Jake Kaplan*

1 Recap from Monday

Hypothesis 1.1 (Expanded Time Hypothesis (ETH))**.** *3SAT takes* $T(n) = 2^{\Omega(n)}$

Definition 1.2 (Approximate **Densest-k-Subgraph (DkS)** Problem)**.** Given an undirected graph G, distinguish between:

Completeness: G has a k-clique;

Soundness: $\forall$ k-subgraphs $S \subset G$, S contains only $o(k^2)$ edges.

Last lecture we saw that DkS is unlikely to be NP-hard:

Theorem 1.3 (previous lecture)**.** *DkS is solvable in quasi-polynomial time*

In particular, NP-hardness of DkS would contradict ETH.

Today we will show that assuming the same ETH, DkS also doesn't admit an actual efficient (polynomial time) algorithm:

Theorem 1.4 (today's lecture)**.** *DkS is not solvable in polynomial time (assuming ETH)*

2 Aside: between P and NP

Before we dive into the details of the proof of hardness for DkS, let's stop for a moment and look at the (much) bigger picture. Over the past several weeks we discussed different types of barriers for proving NP-hardness, and techniques and assumptions that allow us to still reason about the complexity of those problems.

Those barriers can be divided into four categories:

`TFNP (total problems)`

- Classes PPP, PWPP, PPAD, PLS
- Nash equilibrium
- Crypto (CRHF)
- Local max

`Average Case`

- Hardness of PAC-learning
- Planted clique
- Learning with Errors (LWE)

`Unique Games`

- NP-hard to solve exactly
- Is SDP approximation factor tight?
- Max cut, Vertex cover
- Coloring (2-2 Conjecture)

`ETH / Quasi-Poly`

- DkS,
- (ε-best) ε-Nash

Test yourself: Do you understand how the barriers for proving NP-hardness are different? If I gave you a new problem that belongs to one or more of those categories, do you think you could match to the corresponding framework(s)?

3 Hardness of DkS

3.1 Preliminary Reduction Attempts

3.1.1 Attempt 1: The 3Col $\rightarrow$ DkS reduction we used for k-Clique on Monday

Below we let $G^{col} = (V^{col}, E^{col})$ denote the 3-Coloring instance we start our reduction with. We call the vertices of this instance *variables* and the edges *constraints* to distinguish them from the vertices and edges of the new DkS instance.

Step 1: partition the variables into subsets of size $\approx \sqrt{n}$

$$V^{col} = V_1 \cup ... \cup V_k \ \left(l := |V_i| = \frac{n}{k} \geq \sqrt{n}\right)$$

Step 2: construct a vertex for each choice of a subset of variables and partial coloring

$$V^{DkS} = \{u_{i,\sigma_i}\}_{i \in [k], \sigma_i : V_i \rightarrow [3]}$$

Step 3: construct an edge between any two vertices whose corresponding partial assignment does not violate any constraints

$$E^{DkS} : (u_{i,\sigma_i}, u_{j,\tau_j}) \text{ s.t. } i \neq j \wedge (\sigma_i, \tau_j) \text{ satisfy constraints on } V_i \times V_j$$

Birthday repetition Since we picked subsets of variables of size $|V_i| \approx \sqrt{n}$, even if the G^{col} graph is sparse, by *birthday paradox*, we expect most pairs of subsets to have at least one constraint between them.

Notice that since we use size $|V_i| \approx \sqrt{n}$, the size of the new instance is:

$$|V^{DkS}| = \frac{n}{|V_i|} \cdot 3^{|V_i|} \approx 2^{|V_i|} \approx 2^{\sqrt{n}} =: N$$

Now, by ETH we have that the time to solve both problems is at least:

$$T(n) \approx 2^n \approx N^{\log N}$$

This exactly matches our quasi-polynomial time algorithm from Monday! :-)

But the soundness is broken... We sketch a counter example to our soundness claim, namely an example where the new G^{DkS} graph has a dense k-subgraph, even though G^{col} is far from 3-colorable. Let us focus on the first m variable sets, $\boxed{V_1, ..., V_m}, ..., V_k$. A few of the variables in $V_1, ..., V_m$ will have neighbors inside $V_1, ..., V_m$. Fix a valid coloring for these variables and color the remaining variables arbitrarily. There are a lot of vertices that correspond to such choice of variable-subset and coloring; pick k such vertices at random. In expectation, $\frac{1}{m}$-fraction of the pairs of vertices will not have edges because they correspond to the same subset of variables. But for any other pair we only check the few variables with the fixed valid coloring, so those edges will always be present in G^{DkS}. Hence the expected density of the corresponding subgraph is $(1 - 1/m)$, and goes to 1 as we increase m.

3.1.2 Attempt 2: Consider ALL subsets $S \in \binom{V^{col}}{\ell}$

We modify our reduction from Attempt 1 by considering all subsets of size $\ell \approx \sqrt{n}$ rather than a fixed partitioning of the variables.

Step 2: construct a vertex for each choice of a subset of variables and partial coloring This is similar to Attempt 1, but w.r.t. our new family of subsets.

$$V^{DkS} = \{u_{S,\sigma_S}\}_{S \subset V^{col} \text{ st } |S|=\ell, \sigma_S : S \to [3]}$$

Step 2: Constructing the edges How should we generalize the edge construction from the previous attempt? Intuitively, we want to construct an edge between any pair of vertices that could correspond to the same valid coloring of the G^{col} graph. This amounts to requiring that they satisfy all the constraints between the sets of variables, and agree on all colors in the intersection.

$$E^{DkS} : (u_{S,\sigma_S}, u_{T,\tau_T}) \text{ s.t. } \sigma_S, \tau_T \text{ satisfy constraints on } S \times T \text{ and agree on } S \cap T$$

Historical remark: this construction[1] is sometimes referred to as the **FGLSS Graph** [FGL+96].

Intuition In Attempt 1, we constructed a counter-example by restricting to an α-fraction of variables in V^{col} (i.e. take first αk subsets). It corresponded to an α-fraction of vertices in V^{DkS}, since the same subset of variables maps to 3^ℓ vertices in V^{DkS}. By contrast, in Attempt 2, an α-fraction of vertices in V^{col} results in an α^ℓ-fraction of vertices in V^{DkS}. This limits the power of such counter-examples, and in fact allows us to prove a somewhat weaker hardness result for DkS [BKKRW17]. But analyzing this reduction is both messy and ultimately gets stuck at a much weaker soundness parameter. Instead we will see a reduction that is both cleaner and more powerful.

[1] The original FGLSS graph is technically a little different. See footnote in [Man17] for details.

3.2 The actual reduction

Theorem 3.1 ([Man17]). *Assuming ETH, there is no polynomial time algorithm for DkS (distinguishing* 1 *vs* $o(1)$*).*

3SAT vs 3Col We will start our reduction from 3-SAT instead of 3-Coloring. Let's compare our three problems (3-SAT, 3-Coloring, and DkS) on two important parameters: number of variables per constraint, and alphabet size (or number values each variable can take). See Table 3.2 below. For DkS, alphabet size 2 corresponds to choosing for each vertex whether it is in the k-subgraph or not.

Table 1: Constraint vs Alphabet comparison

	3-SAT	3-Col	DkS
Variables per constraint	3	2	2
Alphabet size	2	3	2

3-Coloring was convenient to work with because like DkS it only has two variables (vertices) for each constraint (edge). But we will crucially use the fact that 3-SAT variables can take only two values (True or False), like DkS variables/vertices.

Test yourself: As we present the proof, can you see where we use the small alphabet property?

3.2.1 Construction

$$W := \{\text{1-variable assignments }\} = \{(x_1 = T), (x_1 = F), ..., (x_n = T), (x_n = F)\}.$$

Vertices (all ℓ-tuples)

$$V := \binom{W}{\ell}.$$

Edges (a-la FGLSS graph)

$$(u, v) \in E \iff u \cup v \text{ satisfy all constraints, } u \cap v \text{ consistent}$$

Finally set:

$$k := \binom{n}{\ell}$$

Completeness satisfying assignment $\implies$ k-clique.

3.3 Soundness: Proof Sketch

We will use the following lemma from extremal graph theory:

Lemma 3.2 ([Alo02], generalizing [KST54]). *Every α-dense graph has $\geq \left(\frac{\alpha}{2}\right)^{t^2} N^{2t}$ labeled copies of the bi-clique $K_{t,t}$ (Soundness)*

Goal: Show that if the 3-SAT formula has no satisfying assignment, then the entire G has few copies of $K_{t,t}$. In particular, this means that every k-subgraph has few copies; then, using Lemma 3.2, every k-subgraph is sparse.

We will establish our goal by a clever counting argument. We associate each biclique (L, R) in G with a pair of subsets $A, B \subseteq W$ of one-variable assignments as follows. We let A denote the union of all one-variable assignments in L's vertices, and define B analogously (w.r.t. R). We let $f(A, B)$ denote the set of all bicliques associated with (A, B).

$$f(A,B) := \{(L,R) : (L,R) = K_{t,t}, L \neq R \neq \emptyset, \bigcup_{l \in L} l = A, \bigcup_{r \in R} r = B\} \subseteq \binom{V}{t} \times \binom{V}{t}.$$

Thus to get the total number of bicliques we can sum over the sizes of $f(A, B)$ for all $A, B \subset W$.

$$\begin{aligned} \#\text{ of } K_{t,t} &= \sum_{A,B} |f(A,B)| \\ &\leq 2^{4n} \max_{A,B} |f(A,B)|. \end{aligned} \tag{1}$$

(Where the inequality follows because there are at most 2^{4n} choices of pairs (A, B).)

Our new goal is to bound $\max_{A,B} |f(A, B)|$

3.3.1 Bounding $|f(A, B)|$

We begin with an easy upper bound:

$$\forall\, A, B,\ |A| + |B| \leq 2n,$$

and therefore

$$|f(A,B)| \leq \binom{|A|}{l}^t \binom{|B|}{l}^t \leq \binom{\frac{|A|+|B|}{2}}{l}^{2t} \leq \binom{n}{l}^{2t} = k^{2t}. \tag{2}$$

Let us ignore the 2^{4n} factor we lose in Ineq. (1) for now. Even without it, the bound from (2) barely matches the $\left(\frac{\alpha}{2}\right)^{t^2} k^{2t}$ bound from Lemma 3.2, but for density $\alpha = 2$ — this is even higher than the trivial upper bound of $\alpha \leq 1$. Indeed, we don't expect to get a non-trivial upper bound on the density, since we haven't yet used the premise that the 3-SAT instance is far from satisfiable! But observe also that any (sufficiently large) improvement over the bound from (2) will give a non-trivial upper bound on the density.

We now explain how to obtain the strengthening of (2). We do it by case analysis over three different cases, each corresponding to A, B having many variables from each of the following categories.

Definition 3.3 (Good, bad, and ugly variables)**.** For a fixed choice of $A, B \subset W$ and $i \in [n]$, we say that variable x_i is

good if $(x_i = b) \in A \cap B$ (for some $b \in \{T, F\}$);

bad if $(x_i = T) \in A \wedge (x_i = F) \in A$ (or both in B — but notice that by biclique assumption we cannot have one assignment A and the other assignment in B);

ugly otherwise (aka x_i appears at most once in A and B combined).

We're now ready for our case analysis:

Case ugly Suppose that there many ugly variables (say $\geq \epsilon n$). Each ugly i contributes at most 1 (instead of 2) to $|A| + |B|$. Hence in this case, $|A| + |B| < (2 - \epsilon)n$, which immediately gives an improvement over (2).

Case bad Suppose that there many bad variables (say $\geq \epsilon n$). Consider a bad i such that both $(x_i = T)$ and $(x_i = F)$ appear in A. Then any for $(L, R) \in f(A, B)$, L cannot have the same vertex containing both assignments. This decreases the number of ways to choose L. When there are many bad variables, this decrease is significant enough to improve over (2) and rule out dense subgraphs.

Case good When almost all the variables are good, we can recover from A, B a consistent assignment to almost all the variables. The only way this assignment corresponds to a dense subgraph is if the 3-SAT instance is (almost) satisfiable.

References

[Alo02] Noga Alon. Testing subgraphs in large graphs. *Random Struct. Algorithms*, 2002.

[BKKRW17] Mark Braverman, Young Kun-Ko, Aviad Rubinstein, and Omri Weinstein. Eth hardness for densest-k-subgraph with perfect completeness. *In Proceedings of the Twenty-eight Annual ACMSIAM Symposium on Discrete Algorithms, SODA*, 2017.

[FGL+96] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Interactive proofs and the hardness of approximating cliques. *J. ACM*, 43(2):268–292, 1996.

[KST54] Tamás Kovári, Vera Sós, and Pál Turán. On a problem of k. zarankiewicz. *Colloquium Mathematicum*, 1954.

[Man17] Pasin Manurangsi. Almost-polynomial ratio eth-hardness of approximating densest k-subgraph. *In Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, 2017.