COMS E6998: Fine-grained complexity (Spring'26) January 20, 2026

Lecture 1: Introduction

Instructor: *Josh Alman* Scribes: *Baitian Li, Yuval Efron (2022)*

1 Introduction

Administrative details:

- Instructor: Prof. Josh Alman (`josh@cs.columbia.edu`), CSB 504
- Course site: `josh/fine-grained-complexity`
- Josh's office hours: TBD.
- TA: Baitian Li (`bl3052@columbia.edu`), OH: 3–4PM Fridays CSB 506
- Assignments:
  1. Problem sets: 3 in total.
  2. Scribe notes: Each student (probably in pairs) scribes a single lecture.
  3. Final Project: Survey / Research. The final outcomes of the project are both a write-up and a presentation of the work.

2 Motivation for fine-grained complexity

Let's start with a problem called the *Graph Diameter*.

Graph diameter.

Given. $G = (V, E)$, a simple (unweighted, undirected) graph with n vertices and m edges.

Task. Compute the maximum of $d(u, v)$ over all pairs $u, v \in V$, where $d(u, v)$ is the length of the shortest path in G between u and v.

This algorithmic problem has some intriguing applications in the real world. For example, the connections of network routers can be regarded as a graph, and the diameter of this graph in some sense measures how efficiently nodes can communicate with each other. Perhaps one interesting fact is that, in 1999, the diameter of the WWW was found to be 19.

One simple solution to the problem takes $O(nm)$ time: For each vertex u, take u as the starting vertex and run BFS to get all distances $d(u, v)$. Each BFS takes $O(m)$ time, so it takes $O(nm)$ time in total.

In the *dense graph regime*, i.e., $m = O(n^2)$, the above algorithm only provides an upper bound of $O(n^3)$. There are some better algorithms in the dense regime: Seidel's algorithm [Sei95] employs a clever way to use fast matrix multiplication, and can even compute the whole distance table on a simple graph. It runs in $O(n^\omega \log n)$, where ω is the matrix multiplication exponent (the current bound is $\omega < 2.372$).

But now let's look at the *sparse regime*, which one may think of as $m = O(n)$. In this regime, Seidel's algorithm has no advantage — the fundamental reason is that Seidel's algorithm computes the *entire distance table*[1], so the output inherently has size n^2. Moreover, the trivial BFS solution already runs in $O(n^2)$ time. But since we only care about *one number*, the diameter, *a priori* it might be possible to do better than $O(n^2)$.

Since the $O(nm)$ algorithm is so simple and this problem is so fundamental, experts in graph algorithms have sought better algorithms. But decades later, a better algorithm remains elusive.

Our goal for today's lecture is to prove something along these lines.

This motivates people to consider the possibility in the opposite direction, i.e., we will try to prove that the graph diameter problem is *hard*:

> *There is no algorithm running in $o(n^2)$ time that solves the diameter problem in the sparse regime $m = O(n)$.*

2.1 Attempts to prove hardness

But how can we prove that some problem is *hard*? If we want to prove that the graph diameter problem is hard, we probably want to first identify some known examples for which we know how to prove hardness.

Attempt 1. If we are asked to output all n^2 entries of the distance table, then it requires n^2 time, right?

Answer. Yes, as mentioned before, this is the APSP problem. However, since you are required to output n^2 values, the n^2 lower bound becomes less interesting — it is simply n^2 bounding your total input and output size! That said, the n^2 lower bound for APSP didn't show a *gap* between the time complexity of the algorithm and the size of the input/output, so it didn't actually tell us the true hardness.

Attempt 2. We do know *some* hardness results, right? A Turing machine requires $\Omega(n^2)$ time to decide whether a string is a palindrome. Another example is that sorting requires $\Omega(n \log n)$ time.

Answer. That's true, but caveats apply. To be more precise, the $\Omega(n^2)$ lower bound for palindrome [Maa84] is a strongly *model-dependent* result — it is a lower bound on the *single-tape* Turing machine model. But if you write Python code to decide a palindrome, you will likely obtain a "linear-time" Python algorithm. Indeed, if we slightly change the model of computation to a multi-tape Turing machine, palindrome also admits a linear-time algorithm. For the sorting problem, the famous $\Omega(n \log n)$ lower bound is for the *comparison model*: It restricts how you access the input data.[2]

These examples actually reveal a serious problem: The concept of time complexity itself is also *model-dependent*! Because of this issue, we will mainly focus on the *RAM model*. Although we are not going to define it formally now, broadly speaking, you don't need to worry about memory locality in this model — the time to reach specific memory addresses can be neglected. It captures our intuition of real-world computation and can efficiently simulate most natural computational models that people care about. So

[1] This is called the *all-pairs shortest path (APSP)* problem, which we will study in later lectures.

[2] Indeed, for the sorting problem, there are *some* algorithms faster than $O(n \log n)$: Under a reasonable *RAM model*, you can sort n word-bit integers in $O(n\sqrt{\log \log n})$ time [HT02].

if we really obtain a lower bound on the RAM model, it arguably tells a reasonable lower bound that is not restricted by the model specifics.

Another possible attempt is to start with some source of hardness we know, and reduce such a known-hard problem to the diameter problem.

Well, actually we indeed know some hardness theorems — the *time hierarchy theorem*: $\mathsf{P} \neq \mathsf{EXP}$, i.e., there *are* some problems solvable in exponential time but do not have a polynomial-time algorithm. The statement $\mathsf{P} \neq \mathsf{EXP}$ itself is just made model-independent. Under concrete models, there are some more fine-grained results: $\mathsf{TIME}[o(n^2/\log n)] \subsetneq \mathsf{TIME}[n^2]$, which means that there are some problems solvable in $O(n^2)$ time but not solvable in $o(n^2/\log n)$ time; this applies to both the multi-tape Turing model and the RAM model.

Can we hope to prove the hardness of the diameter problem from a known hard problem in $\mathsf{TIME}[n^2]$? Unfortunately, this is not the case. The time hierarchy theorem is proved via a *diagonalization argument* (a scaled-down version of Turing's proof of the undecidability of the halting problem). So it only shows the hardness of some *universal problems* in $\mathsf{TIME}[n^2]$: The problem is simulating a given program in $n^2/\log n$ steps and outputting the result. We don't think computing the diameter of a graph with $O(n)$ edges can really simulate such a computational process.

Indeed, proving unconditional hardness results beyond the hierarchy theorem is perhaps one of the biggest open questions in computational complexity theory. Not only does the $\mathsf{P} \neq \mathsf{NP}$ conjecture remain elusive, but people currently do not even know how to unconditionally exclude the possibility that any natural NP-complete problem has an $O(n)$ time algorithm.

2.2 Basing hardness on the hardness of SAT

Alas, it seems that we won't be able to prove the hardness of the diameter problem unconditionally, thus we seek some weaker goals: Can we base the hardness of Graph Diameter on some more well-studied problem that is more plausibly hard?

If we are allowed to use the hypothesis $\mathsf{P} \neq \mathsf{NP}$, can we now prove that graph diameter requires $\Omega(n^2)$ time? A natural idea would be to try and design a reduction from an NP-complete problem, e.g., SAT, to graph diameter. But note that this reduction must be very different from what we did in polynomial-time reductions to prove the NP-completeness of problems. Because we already know that graph diameter *can* be solved in $O(n^2)$ time. If one can solve diameter in T time, the reduction should let the SAT algorithm suddenly change from super-polynomial time to polynomial time when T changes from n^2 to, say, $n^{1.9}$. For example, if the reduction implied that one can solve SAT in time $2^{\frac{T}{n^{1.9}}} \cdot \mathrm{poly}(n)$, this would yield that T has to be $\omega(n^{1.9})$, assuming $\mathsf{P} \neq \mathsf{NP}$ (otherwise, SAT would be solvable in polynomial time).

However, such a reduction seems unnatural and is out of reach of current techniques at the very least.

It seems that the assumption that $\mathsf{P} \neq \mathsf{NP}$ is simply too *coarse*. A more *fine-grained* assumption on the hardness of NP-complete problems might allow us to design appropriate reductions. For this, we first review the most fundamental problem in NP — the k-SAT problem.

> **k-SAT.**
> *Given.* A k-CNF formula ϕ on n variables and m clauses.
> *Task.* Decide whether there exists an assignment x satisfying ϕ.

The $\mathsf{P} \neq \mathsf{NP}$ assumption implies that for any $k \geq 3$, there is no polynomial-time algorithm solving k-SAT. However, this lower bound is still very far from the current best-known upper bounds. The state of the art for solving 3-SAT stands at $O(1.31^n)$ by a recent work of [HKZZ19], and for general k-SAT, the state of the art stands at $O(2^{(1-c/k)n})$ where $c = \pi^2/6$, given by [PPSZ05]. (We will learn this in later lectures.) Note that for k-SAT, as k approaches infinity, this algorithm essentially boils down to the naive brute force algorithm for solving SAT.

For this reason, people formulated the following hypothesis, conjecturing that not only does k-SAT require exponential time, but the exponent is essentially not improvable.

Conjecture 2.1 (Strong Exponential Time Hypothesis, SETH)**.** For every $\varepsilon > 0$, there exists a $k \geq 3$, such that k-SAT cannot be solved in $O(2^{(1-\varepsilon)n})$ time.

We are now ready to state a formal, concrete statement regarding the hardness of graph diameter, one we can actually hope to prove.

Theorem 2.1. *Assuming SETH, for any constant $\varepsilon > 0$, there is no $O((n+m)^{2-\varepsilon})$ time algorithm solving the diameter on simple graphs.*

In order to prove the above theorem, we are going to employ an intermediate problem, which we will encounter often throughout the course.

3 Orthogonal vectors

Orthogonal vectors.

Given. Two sets $X, Y \subseteq \{0,1\}^d$, each of size N.

Task. Decide whether there exist $x \in X, y \in Y$ such that $\langle x, y \rangle = 0$, i.e., x, y are disjoint as subsets of $[d]$.

Let's first look at two simple ideas to solve the OV problem. The first one is to brute-force check all pairs, so this takes $O(N^2 d)$ time. Another one is to somehow go over all 2^d possible vectors, which takes $O(N + 2^d \cdot d)$ time.

If d is very small, say $d \leq \lg N$, then the second algorithm runs in nearly linear time. But it quickly becomes inefficient when $d \geq 2 \lg N$.

The OV conjecture asserts that the brute-force algorithm cannot be significantly improved. Actually there are *some* improvements, but the current state of the art is still far from refuting the OV conjecture.

Conjecture 3.1 (OV Conjecture[3])**.** There is no algorithm for OV that runs in $O(N^{2-\varepsilon} \cdot \mathrm{poly}(d))$ for any constant $\varepsilon > 0$.

Now we are going to prove this very fundamental lemma. It was initially proved by Williams [Wil05], and it turned out to be the starting point of fine-grained complexity.

Lemma 3.1. *SETH implies the OV conjecture.*

[3]The precise definition of the OV conjecture is actually slightly different from what we introduced in this lecture, but the current definition is sufficient for our purpose (basing the hardness of graph diameter on SETH, where the OV conjecture plays the role of a springboard).

We will prove this by giving a *fine-grained reduction* from k-SAT to OV: We will reduce an n-variable k-SAT problem to sets of size $N = 2^{n/2}$. This is very different from traditional reductions that are polynomial time. In fine-grained reductions, the reductions might not be polynomial, but we need to carefully keep track of the blow-up.

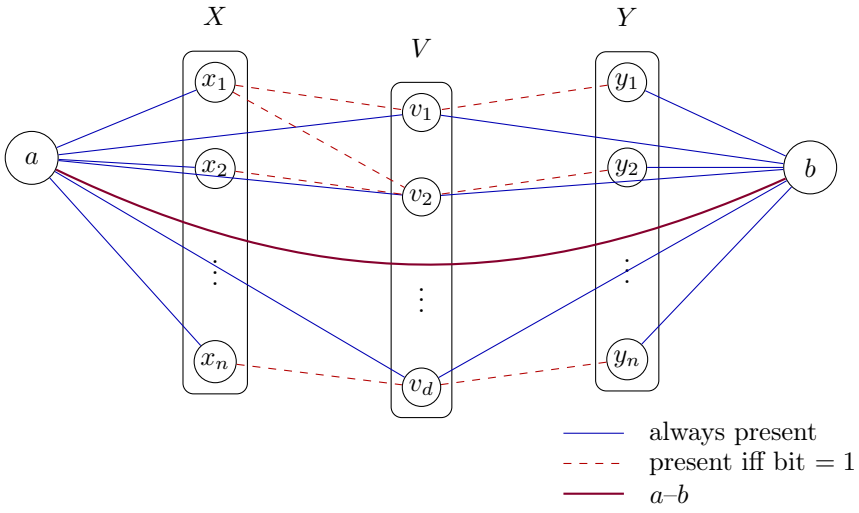
Proof. Consider the k-CNF formula $\phi = \psi_1 \wedge \cdots \wedge \psi_m$ in n variables. It should have at most $m = O(n^k)$ essentially different clauses because every clause has at most k variables.

Denote the variables of ϕ by $x_1, \ldots, x_n$ and let $S_1 = \{x_1, \ldots, x_{n/2}\}$ and $S_2 = \{x_{\frac{n}{2}+1}, \ldots, x_n\}$. For each $i \in \{1, 2\}$, let A_i be the set of assignments to all variables in S_i. Namely $A_i = \{\alpha : S_i \to \{\texttt{True}, \texttt{False}\}\}$. Note that $|A_i| = 2^{n/2}$.

For each $\alpha \in A_i$, regarded as a partial assignment of ϕ, it might or might not satisfy each clause ψ_j. Construct a vector $v_\alpha \in \{0, 1\}^m$ such that $v_\alpha[j] = 0$ if α satisfies the clause ψ_j, and 1 otherwise. Denote $X = \{v_\alpha \mid \alpha \in A_1\} \subseteq \{0, 1\}^m$ and $Y = \{v_\beta \mid \beta \in A_2\} \subseteq \{0, 1\}^m$. It is not too hard to see that there exist $x \in X$ and $y \in Y$ that are disjoint as subsets of $[m]$ iff ϕ is satisfiable.

Assume the OV conjecture is false, i.e., that we can solve OV in time $O(N^{2-\varepsilon} \cdot \text{poly}(d))$ for some constant $\varepsilon > 0$. Since $N = 2^{n/2}$ and $m = \text{poly}(n)$, our reduction solves k-SAT with time complexity bounded by $O(2^{(1-\varepsilon/2)n} \text{poly}(n)) \subset O(2^{(1-\varepsilon/3)n})$. This falsifies SETH. □

Lemma 3.2. *Under the OV conjecture, there is no $O((n+m)^{2-\varepsilon})$ time algorithm solving the diameter problem for any constant $\varepsilon > 0$.*



Proof. We prove this by presenting a reduction from OV to Graph Diameter. Given an instance of OV, denoted by $X, Y \subseteq \{0, 1\}^d, |X| = |Y| = n$, where $X = \{x_1, \ldots, x_n\}, Y = \{y_1, \ldots, y_n\}$. We construct a graph $G = (V, E)$ in the following way.

$$V = \{x_i \mid x_i \in X\} \sqcup \{y_i \mid y_i \in Y\} \sqcup \{v_1, \ldots, v_d\} \sqcup \{a, b\}$$

The edges of the graph are given as follows:

- Connect (x_i, v_j) if $x_i[j] = 1$, and similarly for (y_i, v_j) for all possible pairs.

- Connect a to all of $v_1, \ldots, v_d$ and $x_1, \ldots, x_n$, and similarly b to all of $v_1, \ldots, v_d$ and $y_1, \ldots, y_n$.
- Finally, we connect a to b.

One can then check the following facts:

- The diameter of the graph is at most 3.
- If x, y is not an $X \times Y$ pair (up to swapping), then their distance $d(x, y) \leq 2$.
- If x, y is an $X \times Y$ pair, then $d(x, y) = 2$ if they are not an orthogonal pair (path through where they intersect); otherwise, $d(x, y) = 3$ (path through a, b).

In conclusion, (X, Y) is an OV yes-instance iff the graph diameter is 3.

Assume graph diameter can be solved in $O((|V|+|E|)^{2-\varepsilon})$ time. The above reduction has $|V| = O(n+d)$ and $|E| = O((n+d)d)$. This solves OV in $O(n^{2-\varepsilon}\,\mathrm{poly}(d))$ time, thus refuting the OV conjecture. □

Thus we can conclude from Lemmas 3.1 and 3.2 the desired Theorem 2.1.

References

[HKZZ19] Thomas Dueholm Hansen, Haim Kaplan, Or Zamir, and Uri Zwick. Faster k-sat algorithms using biased-ppsz. In *STOC*, pages 578–589. ACM, 2019.

[HT02] Yijie Han and Mikkel Thorup. Integer sorting in 0(n sqrt (log log n)) expected time and linear space. In *43rd Symposium on Foundations of Computer Science, FOCS 2002, Vancouver, BC, Canada, November 16-19, 2002, Proceedings*, pages 135–144. IEEE Computer Society, 2002.

[Maa84] Wolfgang Maass. Quadratic lower bounds for deterministic and nondeterministic one-tape turing machines. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 401–408, 1984.

[PPSZ05] Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. An improved exponential-time algorithm for k-sat. *J. ACM*, 52(3):337–364, 2005.

[Sei95] Raimund Seidel. On the all-pairs-shortest-path problem in unweighted undirected graphs. *J. Comput. Syst. Sci.*, 51(3):400–403, 1995.

[Wil05] Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoret. Comput. Sci.*, 348(2-3):357–365, 2005.