CS 0368-4246: Combinatorial Methods in Algorithms (Spring 2025) April 7, 2025

Lecture 4: Cycle Finding and Lower Bounds for Approximation Algorithms

Instructor: *Or Zamir* Scribes: *Yahav Boneh*

1 Introduction

In the previous lecture, we discussed lower bounds for the complexity of algorithms under certain hardness assumptions. Particularly, we only considered algorithms that give *exact* solutions. In this lecture, we will show results on *approximation* algorithms, and also bound the complexity of cycle finding.

2 Cycle Finding Complexity

We saw several algorithms that find cycles; C_{2k} in $O\left(n^2\right)$, C_4 in $O\left(m^{\frac{4}{3}}\right)$, C_{2k+1} in n^ω, and C_3 in $m^{\frac{2\omega}{1+\omega}}$. The question that arises naturally is: are the cycles of all lengths equivalent?

Claim 1. *Finding a triangle in a graph is equivalent to finding a triangle in a 3-partite graph.*

Proof. One direction is trivial. For the other, given a graph $G = (V, E)$, consider a graph on the vertices $V \sqcup V' \sqcup V''$ where we connect $(i, j'), (i, j''), (i', j), (i', j''), (i'', j), (i'', j')$ for every edge $(i, j) \in E$. It is clear that G contains a triangle if and only if the new graph does, and the new graph is 3-partite. □

Claim 2. *Given an algorithm for finding C_{2k+1} in $O\left(m^{1+o(1)}\right)$ time, we can find C_3 in $O\left(m^{1+o(1)}\right)$ time too.*

Proof. We may assume, without loss of generality, that we are looking for a triangle in a 3-partite graph. Let $G = (A \sqcup B \sqcup C, E)$ be that graph. In order to translate triangles to cycles of length $2k + 1$, replace each edge that goes between B and C by a P_{2k-1}. If there was a triangle in the original graph, we must have replaced one of its edges by P_{2k-1}, making it C_{2k+1}. For the other direction, let N_i $(1 \leq i \leq 2k - 2)$ be the set of all new vertices which are i edges into the new P_{2k-1}, where we start counting from B. Clearly, there are no edges in any N_i, and our graph has edges only between $C \leftrightarrow A \leftrightarrow B \leftrightarrow N_1 \leftrightarrow N_2 \leftrightarrow \cdots \leftrightarrow N_{2k-2} \leftrightarrow C$. Therefore, if we remove any single set out of $A, B, C, N_1, N_2, \ldots, N_{2k-2}$, we are left with a bipartite graph. Bipartite graphs can't have cycles of odd length, which means that every C_{2k+1} in the new graph must go through all of $A, B, C, N_1, N_2, \ldots, N_{2k-2}$, so it is in correspondence with a triangle in the original graph. □

The same argument won't work for even cycles, as we won't be left with a bipartite graph at the end. Nevertheless, we can use the following claim for even cycles.

Claim 3. *Given an algorithm for finding $C_{k \cdot l}$ in $O\left(m^{1+o(1)}\right)$ time, we can find C_k in $O\left(m^{1+o(1)}\right)$ time too.*

Proof. Replace each edge with P_l. It is clear that a $C_{k \cdot l}$ in the new graph is in correspondence with a C_k in the original one. □

These two claims immediately imply the following theorem.

Theorem 4. *If we cannot find C_3 in $O\left(m^{1+o(1)}\right)$ time, then we cannot find C_k in $O\left(m^{1+o(1)}\right)$ time for any k which is not a power of 2. If we cannot find C_4 in $O\left(m^{1+o(1)}\right)$ time, then we cannot find C_k in $O\left(m^{1+o(1)}\right)$ time for any $k = 2^r$ $(r > 1)$.*

3 Approximation Algorithms

The methods we saw earlier don't generalize to approximation algorithms. When we build a new graph as before, it is built specifically for the length of the cycle we want to find. If we look for a different length, we might find one that does not correspond to the original graph anymore.

3.1 Short Cycle Removal [1]

Intuitively, we will show that every graph G can be quickly transformed into a graph G' of similar size, such that

1. $C_3 \in G' \iff C_3 \in G$.
2. For every $k' \in [4, k]$ (where k is a constant), $C_{k'} \notin G'$.

Before making this intuition formal, we establish some more hardness assumptions.

Assumption 5 (Weak Triangle)**.** *No algorithm can find C_3 in a graph in $O\left(m^{1+o(1)}\right)$ time.*

Assumption 6 (Strong Triangle)**.** *No algorithm can find C_3 in a graph with $\max_{v \in V} d(v) \leq \sqrt{n}$ in $O\left(n^{2-\varepsilon}\right)$ time.*

Assumption 7 (All Edge Triangle)**.** *No algorithm can decide for every edge if it is a part of a C_3 in $O\left(n^{2-\varepsilon}\right)$ time.*

The last assumption becomes much more plausible with the following theorem, which we do not prove.

Theorem 8. *If APSP or 3-Sum are correct, then All Edge Triangle is correct too, even if $\max_{v \in V} d(v) \leq \sqrt{n}$ and $\#C_3 \leq n^{1.6}$.*

Moving forward, we will mainly use the All Edge Triangle assumption, and assume that $\max_{v \in V} d(v) \leq \sqrt{n}$ and that $\#C_3 \leq n^{1.6}$. Such graphs have $m \leq n^{\frac{3}{2}}$, and $\#C_k \leq n \cdot (\sqrt{n})^{k-1} = n^{\frac{k+1}{2}}$. The latter bound is still large, and even for $k = 4$, we possibly get $\#C_4 = n^{2.5} >> n^2$. We would like to decrease the number of C_4s, or C_ks in general, to $n^{2-\varepsilon}$.

Theorem 9. *Let $k \geq 4$, $\alpha \in \left(0, \frac{1}{2}\right)$ and $\varepsilon \in \left(0, \frac{3-\omega}{4}\right)$. Let G be a graph with maximal degree $\sqrt{n}$. It is possible, in $O\left(n^{2-\varepsilon}\right)$ probabilistic time, to find a subset $E' \subseteq E$ and a sequence of sub-graphs $G_1, \ldots, G_s \subseteq G$ such that:*

1. Each edge in E' is a part of a C_3 in G.

2. *Every edge in E which is a part of a C_3 in G is either in E' or in some G_i.*

3. *$s \leq n^{\frac{3}{2}-3\alpha}$, every graph G_i has at most $n^{\frac{1}{2}+\alpha}$ vertices and maximal degree at most n^{α}.*

4. *For every $4 \leq k' \leq k$, each G_i has at most $n^{\frac{\omega-1}{4}+k\alpha+\varepsilon}$ cycles of length k'.*

The total number of vertices in the new graphs is $N \leq n^{\frac{3}{2}-3\alpha} \cdot n^{\frac{1}{2}+\alpha} = n^{2-2\alpha}$, the total number of edges is $M \leq n^{\frac{3}{2}-3\alpha} \cdot n^{\frac{1}{2}+\alpha} \cdot n^{\alpha} = n^{2-\alpha}$, and the total number of $C_{k'}$s ($k' \in [4,k]$) is $\#C_{k'} \leq n^{\frac{3}{2}-3\alpha} \cdot n^{\frac{\omega-1}{4}+k\alpha+\varepsilon} = n^{\frac{3}{2}+\frac{\omega-1}{4}+(k-3)\alpha+\varepsilon}$. If we choose a small enough α (dependent on k) and a small enough ε, using the fact $\omega < 3$, there exists some $\delta > 0$ such that $N, M, \#C_{k'} \leq n^{2-\delta}$, where $\delta \approx \frac{1}{k}$.

This theorem is not as strong as the intuition described in the beginning of the section; however, we will see it is still usefull.

Claim 10. *Under APSP/3-Sum/All Edges, there is no algorithm that finds t C_4s in $m^{1+o(1)} + tm^{o(1)}$ time.*

Proof. Assume towards contradiction that such algorithm exists. We will show a reduction from All Edges. Let G be a graph (with $\max_{v \in V} d(v) \leq \sqrt{n}$ and $\#C_3 \leq n^{1.6}$). Apply Theorem 9 with $k = 4$. For every G_i, apply the reduction we saw earlier (transform edges two of the three parts to P_2). On every G_i, run the algorithm (that we assume exists) to find all traingles in G_i in time $m^{o(1)}\left(\#C_4\left(G_i\right) + \#C_3\left(G_i\right)\right)$. In order to verify whether an edge actually belongs to a triangle, we will check in $O\left(n^{\alpha}\right)$ time. Our total time complexity after running the first transformation is the sum of:

1. $m^{1+o(1)}$ for every G_i, which sums to $M^{1+o(1)} < n^{2-\delta}$.
2. All of the preexisting C_4s times n^{α} — less than $n^{2-\delta}$ because there weren't many C_4s.
3. All of the preexisting C^3s times n^{α} — also less than $n^{2-\delta}$ as $\#C_3 \leq n^{1.6}$.

In conclusion, we have reached a contradiction to the All Edge hypothesis. □

Exercise 11 (Distance Oracle)**.** *Prove that there is no algorithm that can, after $O\left(m^{1+\frac{1}{100k}}\right)$ time preprocessing, answer (u,v)-queries in time $O\left(m^{\frac{1}{100k}}\right)$, where the answer to the query is the distance between u and v up to a k-factor (the answer should be at least the real distance and at most k times the real distance).*

We provide the proof sketch. start from All Edges, apply Theorem 9 to get $G_1, \ldots, G_s$. In every G_i (assumed to be 3-partite), remove all the edges between two of its parts. Apply the (assumed existing) algorithm and query each of the removed edges. If the distance if 2, the algorithm won't return more than $2k$. The main claim is that the algorithm won't give a false answer too many times.

We proceed to prove Theorem 9.

Proof of Theorem 9. The first idea is that if we create a subgraph of G by keeping each vertex with probability p, then the number of C_3s gets p^3 times smaller, while the number of C_4 gets p^4 times smaller, significantly less. This idea is formulated in this lemma.

Lemma 12. *It is possible to transform a graph G with t C_4s to $G_1, \ldots, C_s$ with our desired sizes and degress, such that the number of C_4s is now $\leq \frac{t}{\sqrt{n}} \cdot n^{\alpha}$, such that every triangle of G still apears in some G_i.*

Proof. Without loss of generality, G is a 3-partite graph. Divide each part randomly into $n^{\frac{1}{2}-\alpha}$ subparts. For every three subparts (each from a different main part), create G_i for the induced subgraph on these vertices. We have $s \leq \left(n^{\frac{1}{2}-\alpha}\right)^3 = n^{\frac{3}{2}-3\alpha}$. The number of vertices in each G_i is $|V(G_i)| \leq \frac{n}{n^{\frac{1}{2}-\alpha}} = n^{\frac{1}{2}+\alpha}$. The maximal degree in G_i (expected value) is $\deg_{G_i} \leq \frac{\sqrt{n}}{n^{\frac{1}{2}-\alpha}} = n^{\alpha}$. By Chernoff, if we would multiply the upper bound by $\log n$, the probability that it won't be the case would be $\leq \frac{1}{n^{100}}$. We also have $\#C_4 \leq t \cdot \frac{1}{n^{\frac{1}{2}-\alpha}} = \frac{t}{\sqrt{n}} \cdot n^{\alpha}$, as we need two vertices from the same part to be on the same subpart. □

The problem is that we may have $t = n(\sqrt{n})^3 = n^{2.5}$ C_4s, in which case our reduction will give us $\frac{n^{2.5}}{n^{0.5}} \cdot n^{\alpha} = n^{2+\alpha} >> n^2$ C_4s, which doesn't help us.

Note: the proof continues only for $k = 4$ due to time considerations in the lectures.

Our goal is now to take a G under our assumptions and decrease the number of C_4s to $n^{2.49}$. We will show that every graph with "too many" C_4s, there must be a dense subgraph (which we can find efficiently), and then show that we can answer All Edges problems fast enough for dense graphs. Let $\gamma = \frac{\omega-1}{4} + \varepsilon$. We call a subgraph with $\leq 2\delta n$ vertices "dense" if it has at least $n^{\frac{1}{2}+\gamma}$ edges. We say that an edge is dense if it is contained in at least $n^{\frac{1}{2}+\gamma}$ C_4s.

Lemma 13. *Given a dense edge, we can find a dense subgraph. It can be done efficiently, and we can efficiently check that it is dense.*

Proof. The number of C_4s containing (u, v) is the number of edges in $N(u) \times N(v)$ (up to a possible factor of 2). Therefore, if (u, v) is dense, $N(u) \cup N(v)$ is dense. □

In order to check if an edge is dense, we will sample $\frac{n}{n^{\frac{1}{2}+\gamma}} \cdot \log n$ possible edges between $N(u)$ and $N(v)$ and check how many exist.

Lemma 14. *If $\#C_4 > 10n^{2+\gamma}$, there exists at least $n^{1+\gamma}$ dense edges.*

Proof. We have,

$$\begin{aligned} 10n^{2+\gamma} &< \#C_4 < \sum_e \#\{C_4\text{s using e}\} \\ &\leq \underbrace{n^{\frac{3}{2}}}_{\substack{\text{number of non-dense edges} \\ \leq \text{ number of edges}}} \cdot \underbrace{n^{\frac{1}{2}+\gamma}}_{\text{not dense}} + \underbrace{t}_{\text{number of dense edges}} \cdot \underbrace{n}_{\substack{\text{maximal amount of } C_4\text{s} \\ \text{going through an edge}}} \end{aligned} \tag{1}$$

Therefore, $t > n^{1+\gamma}$. □

Corollary 15. *A randomly sampled edge is dense with probability $\frac{n^{1+\gamma}}{n^{1+\frac{1}{2}}} = \frac{1}{n^{\frac{1}{2}-\gamma}}$.*

Hence, if will sample edges and test if they are dense, we will find a dense edge in $n^{1-2\gamma}$ time.

We are left to solve All Edges on the edges of a dense subgraph H. Fix β which will be chosen later. We divide the neighbors of H (the vertices outside of H which are connected to H) to two parts:

- V_h: vertices with $\geq n^{\frac{1}{2}-\beta}$ edges into H.
- V_l: vertices with $< n^{\frac{1}{2}-\beta}$ edges into H.

For every vertex in V_l, simply test for every two edges going from it to H if they close a triangle. If the degree of the vertex is d, it takes d^2 time. The number of vertices of degree d into H is at most $\frac{n}{d}$, so testing all vertices in V_l of degree d into H will take (together) $\frac{n}{d} \cdot d^2 = nd$. Therefore, testing all the vertices in V_l will take $n \cdot n^{\frac{1}{2}-\beta} = n^{\frac{3}{2}-\beta}$ time in total. We have $|V_h| \leq \frac{n}{n^{\frac{1}{2}-\beta}} = n^{\frac{1}{2}+\beta}$. Using fast matrix multiplication, we can test them in time $n^\beta \cdot (\sqrt{n})^\omega = n^{\frac{\omega}{2}+\beta}$ (divide the $n^{\frac{1}{2}+\beta}$ vertices to n^β subsets of size $\sqrt{n}$, and run the matrix multiplication on them). There is a value for $\beta > 0$ for which both $\frac{3}{2} - \beta < \frac{3}{2}$ and $\frac{\omega}{2} + \beta < \frac{3}{2}$.

In $n^{1-2\gamma}$ time, we found a subgraph with $\geq n^{\frac{1}{2}+\gamma}$ edges, and checked if each of them is in a C_3 in $n^{\frac{3}{2}-\gamma}$ time. On average, every edge took $\frac{n^{\frac{3}{2}-\gamma}}{n^{\frac{1}{2}+\gamma}} = n^{1-2\gamma}$ time. Alltogether, we couldn't have done it for more than $\frac{n^{\frac{3}{2}}}{n^{\frac{1}{2}+\gamma}} = n^{1-\gamma}$ edges, so it took us a total of $n^{2-3\gamma}$ time. In time $<< n^2$, we answered for a part of the edges, while reducing the number of C_4s to $<< n^{2.5}$, from which point we can use the random reduction. □

Exercise 16. *Finish the proof for C_k.*

References

[1] Amir Abboud, Karl Bringmann, Seri Khoury, and Or Zamir. Hardness of approximation in p via short cycle removal: cycle detection, distance oracles, and beyond. In Stefano Leonardi and Anupam Gupta, editors, *STOC 2022 - Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, Proceedings of the Annual ACM Symposium on Theory of Computing, pages 1487–1500, USA, September 2022. Association for Computing Machinery. Publisher Copyright: © 2022 ACM.; 54th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2022 ; Conference date: 20-06-2022 Through 24-06-2022.