CS 0368-4246: Combinatorial Methods in Algorithms (Spring 2025) March 31, 2025

Lecture 3: Cycle Finding and Lower Bounds for Algorithms

Instructor: *Or Zamir* Scribes: *Ilai Segev*

1 Introduction

In previous lectures, we explored probabilistic algorithms for finding simple structures in graphs, such as paths and cycles. In this lecture, we continue this theme by introducing randomized methods for detecting cycles of even length, particularly C_{2k}. We also begin our exploration of fine-grained complexity and conditional lower bounds, based on widely believed conjectures such as the Strong Exponential Time Hypothesis (SETH).

2 Cycle Finding

Theorem 1. *For any fixed k, it is possible to determine in $O(n^2)$ whether the graph contains C_{2k}.*

Lemma 2. *Let there be a connected graph with at least $2tn$ edges, where the vertices are colored with at least three colors. Then, the graph must contain a path P_t whose endpoints are colored with different colors.*
If the graph is not bipartite, then two colors are sufficient.
Moreover, such a path can be found in time $O(m)$.

Proof. Since $m > tn$ the graph contains a θ-graph with girth $> t$, therefore such a graph can be found in time $O(m)$, denote it H.
It's possible to check in time $O(n)$ whether H is t-cyclic.
Why? Each vertex "initiates" at most 8 simple paths of length t in H. We first transfer the graph to an array and then check in time complexity of $O(1)$ for each vertex.
If H isn't t-cyclic, we're done (we found a path).
Assuming it is, by the lemma from last week, H is 2-cyclic. Inside H there are only two colors. Take a vertex v in the graph that is colored with a different color than the two in H.
Find the shortest path (using BFS) from v to H. This path is simple and touches H only at its endpoint.
Extend the simple path on H by more than t steps such that the total length is divisible by t.
Now the path "breaks" into a sequence of simple paths of length t, where the first and last vertices are colored differently. So at least one of these paths must "switch color".
Now assume also that G is not bipartite. If we used 3 or more colors, we finished as before.
Otherwise, G is colored with 2 colors, then there must be an edge whose endpoints share the same color, denote this edge (u, v).
If H is colored with a single color we're done (as in the previous case). Otherwise, we'll find a path from u to H. We may assume it does not pass through v, since at least one of them has a path to H that doesn't go through the other.

We extend the path to obtain a path from $v$to a vertex on H with length divisible by t. If we add one more edge on the cycle, we get such a path from u.
Either one of the paths we found is path P_t whose endpoints are colored with different colors, or there are two adjacent vertices in H with the same color, but this contradicts our assumption that H is 2-cyclic. □

Proof of Theorem.

Algorithm 3. *Takes as input a graph G and vertex v.*
In time $O((k+2)! \cdot n)$ checks whether there exists any C_{2k} in G, or whether there is no C_{2k} passing through v.
Once we have such an algorithm, we can run it on every vertex in the graph.
** Start a BFS traversal from v and during the run maintain the following:*

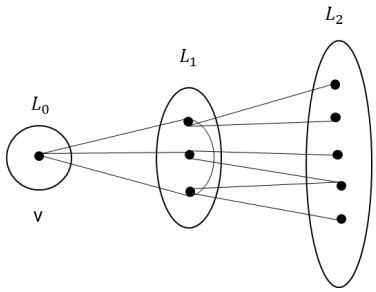

Figure 1: BFS Traversal

- *The layers L_i and their sizes.*
- *The edges within each L_i and how many there are.*
- *The edges between L_i and L_{i+1} and how many there are.*

We stop the BFS if any of the following occurs:

1. *We have finished traversing over the graph, or over the layer L_{k-1}.*
2. *For some i, $4k|L_i| < e(L_i)$.*
3. *For some i, $4k(|L_i| + |L_{i+1}|) < e(L_i, L_{i+1})$.*

What do we do in each stopping case?

1. *If there exists a C_{2k} that passes through v, then we must have seen all its edges. How many edges did we collect? $O(kn)$*
   *We learned how to find in lecture 2 in time $O((k+2)! \cdot m')$: * Checking Whether there exists a C_{2k} that touches v. * All the vertices reachable from v by P_{2k-1}.*
2. *Note that if $4k|L_i| < e(L_i)$ then there exists a connected component within L_i that satisfies this condition. We restrict ourselves to it, and denote it by L'_i.*
   Assuming that L'_i isn't bipartite, We'll find the LCA (lowest common ancestor) of the vertices in L'_i in the BFS graph.

How? *We'll take L_i and build a set of their neighbors in L_{i-1}, then their neighbors in L_{i-2}, and continue this process until the first time we get a set of size one.*
Call the resulting vertex u. The vertex u has more than one child in the BFS tree restricted to L'_i.
We can now color the vertices of L'_i in two colors, one for the descendants of the first child a, and the other for the descendants of the other children.
Now all that's left is to find in L'_i a simple path of length $2k - 2\,(i - depth\,(a))$ between two vertices of different colors.
Using the paths from u, we can complete this into a cycle C_{2k}.
** If L'_i is bipartite, denote $L'_i = K'_i \cup R'_i$.*
We proceed similarly: instead of finding the LCA of all of L'_i we find the LCA u of just one side of the bipartite, say K'_i.
We color the vertices in K'_i with 2 colors as before, and assign a third color to all vertices in R'_i.
By the lemma, there exists a path P_t for $t = 2k - 2\,(i - depth\,(u))$ that connects two vertices of different colors within L'_i. If the endpoints are both from the two different colors in K'_i, we're done.
This must be the case: a path of even length in a bipartite graph must start and end on the same side, so such a path can't connect K'_i, R'_i if it's even length.

3. *Same as above - $K'_i = L'_i$ (the closer set).*

□

Theorem 4 (Bondy-Simonovits)**.** $ex\,(C_{2k}, n) \leq 100kn^{1+\frac{1}{k}}$

Proof. **When does the algorithm fail to find a C_{2k}?** Only case 1 might return false.
We will use the given bound to reduce the graph to a subgraph G' with minimum degree of at least $100kn^{1/k}$.
We run the algorithm on G' and we will show that it cannot stop because of case 1. Therefore, it must return C_{2k}.
Assume by contradiction that condition 1 holds.
That means we built the layers $L_0, ..., L_{k-1}$ (and possibly a part of L_k), and neither condition 2 nor 3 occurred.
From this, we know that for every $1 \leq i \leq k-1$ the number of edges touching the vertices of L_i satisfies

$$12k\,(|L_i| + |L_{i-1}| + |L_{i+1}|) \geq L_i$$

But we also have a lower bound on this quantity due to the minimum degree $|L_i|\,100kn^{1/k}$
From this we can deduce:

$$|L_{i+1}| + |L_{i-1}| \geq |L_i| \left(n^{1/k} \cdot \frac{100 - 1}{12} \right)$$

Using this recurrence, we can prove by induction:

$$|L_1| + \ldots + |L_i| > 2n^{i/k}$$

In particular, $|L_k| > 2n$, which is a contradiction, since the number of vertices in the graph is n. □

3 Lower Bounds for Algorithms

We currently do not know how to prove nontrivial lower bounds on the running time of specific problems. Therefore, we focus on alternative approaches:

- Lower bounds under restricted computational models, e.g., comparison-based sorting
- Lower bounds under assumptions/hypotheses, like reductions in complexity theory.

3.1 Fine-Grained Complexity

We aim to take the concept of reductions from complexity theory,
for example: SAT is not in P $\Rightarrow$ Graph-Coloring is not in P
But make the conclusions more "fine-grained", distinguishing between problems that are all solvable in polynomial time, yet with different polynomial complexities. * This requires finer reductions, but also finer conjectures.
What kinds of hypotheses are even reasonable to consider?
Previously, we had: SAT is not in P.
We can refine it to: SAT requires exponential time.
Even further to: SAT takes C^n for some constant C.

3.2 SAT

- Naive algorithm: $O^*(2^n)$ where O^* hides subexponential factors (Cook, 1971).
- k-SAT
  If $k = 2$ solvable in polynomial time (even linear).
  If $k \geq 3$ NP-Complete (Karp, 2009).
- For general SAT, no algorithm is currently known that beats $O^*(2^n)$.
- For k-SAT, we do know algorithms with running time $(2 - \varepsilon_k)^n$ for some $\epsilon_k > 0$ (Monien and Speckenmeyer, 1985) (Hansen et al., 2019)

In all of these works: $\varepsilon_k \xrightarrow[k\to\infty]{} 0$

3.3 Hypotheses – (Impagliazzo, Paturi, and Zane, 2001)

1. ETH (Exponential Time Hypothesis): SAT or 3-SAT takes C^n for some C.
2. SETH (Strong ETH): $\forall \varepsilon > 0 \exists k \; s.t. \; k - SAT \notin \text{Time}\left((2 - \varepsilon)^n\right)$

3.4 Popular Hypotheses

1. SETH
2. $APSP \notin n^{3-\varepsilon}$ for any $\epsilon > 0$.
3. Triangle Detection:
   - Weak version: cannot be solved in near-linear time $m^{1+o(1)}$.
   - Strong version: cannot be solved in time better than $n^{2-\varepsilon}$ when $m \leq n^{3/2}$

3.5 Reductions from SETH to Problems in P

Problem: Orthogonal Vectors (OV)

- Input: Two sets $A, B \subseteq \{0,1\}^d$ whith $|A|, |B| \leq n$.
- Output: Determine whether there exist vectors $a \in A, b \in B$ such that they are orthogonal:

$$\langle a, b \rangle = \sum_{i=1}^{d} a_i b_i = 0$$

- Algorithms
  - Naive solution: Try all n^2 pairs, $O(n^2 d) \approx n^{2+o(1)}$
  - In time $n2^d$ - insert one of the sets into a hash table. For each vector in the other set, check whether any orthogonal vector exists in the table.
- Assuming $d = \omega(\log n)$, typically $d = \text{polylog(n)}$

Theorem 5. *Under SETH, OV cannot be solved in time $n^{2-\varepsilon}$ for any $\epsilon > 0$ when $d = polylog(n)$*

Proof. Let the input be a k-CNF formula with N variables and M clauses.
Denote the variables as: $x_1, ..., x_N$. Divide them into two halves:

$$x_A = \{x_1, \ldots x_{N/2}\}, x_B = \{x_{N/2+1}, \ldots x_N\}$$

We will construct two sets of vectors A, B such that: $n = 2^{N/2}$, $d = \text{polylog(n)}$.
For each assignment to the variables in x_A, create a vector of length M, coordinate i is 0 iff the assignment satisfies clause i on its own.
Similarly, build vectors for B.

Claim 6. *There exist $a \in A, b \in B$ such that*

$$\langle a, b \rangle = 0 \Leftrightarrow \textit{There exists a satisfying assignment for the formula.}$$

Proof. $\Rightarrow$ Assume a corresponds to an assignment for variables in x_A and b corresponds to variables in x_B. Together, they define an assignment to all variables.
The assignment satisfies, because for any clause the relevant coordinate is 0 either for a or b, since

$\langle a, b\rangle = 0$.
Now assume we have a satisfying assignment for the formula. This assignment corresponds to a unique pair (a, b) of the reduced assignment of x_A, x_B.
We claim that $\langle a, b\rangle = 0$.
For each clause i the full assignment satisfies it. Thus, at least one variable in it is 1, which is in x_A or x_B. Therefore, $a_i = 0$ or $b_i = 0$. □

We've constructed an instance of the OV problem such that solving it gives the answer to the original k-SAT problem.
Where $n = 2^{N/2}$ and $d = M = O(N^k) = O(\log^k n)$. □

Note: *There exists a theorem in complexity theory called the Sparsification Lemma, which says that any k-SAT formula can be rewritten with only $O(N)$ clauses.*

3.6 Another Example - Diameter

Given an undirected and unweighted graph G, compute its diameter: the longest shortest path in the graph $\max_{u,v\in V^2} d_G(u, v)$.
Naive solution: Assume that $m = n^{1+o(1)}$, The problem can be solved by running BFS from each vertex, resulting in an overall time complexity of $O(n^{2+o(1)})$.

Theorem 7. *Under SETH, there is no algorithm that computes the diameter of an unweighted graph with $m \le n^{1+o(1)}$ in time $O(n^{2-\epsilon})$ for any $\epsilon > 0$.*

Proof. We reduce from OV, constructing graph G as the following figure:

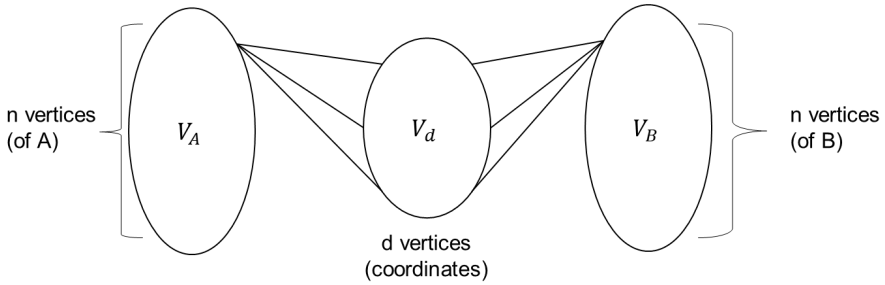

Figure 2: Graph G

Let $A, B \subseteq \{0, 1\}^d$, We construct vertex sets V_A, V_B, V_d corresponding to the vectors in A, B and the coordinates d. For each $a \in A$ the corresponding Vertex in V_A is connected to all vertices in V_d that correspond to coordinates where a has a 1. Similarly, for each $b \in B$
Key Observation: for each $a \in V_A$, $b \in V_B$

$$d(a, b) = \begin{cases} 2 & \langle a, b\rangle \neq 0 \\ > 2 & \langle a, b\rangle = 0 \end{cases}$$

Why? a, b share at least one neighbor iff $\langle a, b\rangle \neq 0$.
If $\langle a, b\rangle = 0$ then $d(a, b) \ge 4$ because G is a bipartite graph.

We'll add Vertices $\{x, y\}$ such that $V = V_A \cup V_B \cup V_d \cup \{x, y\}$:

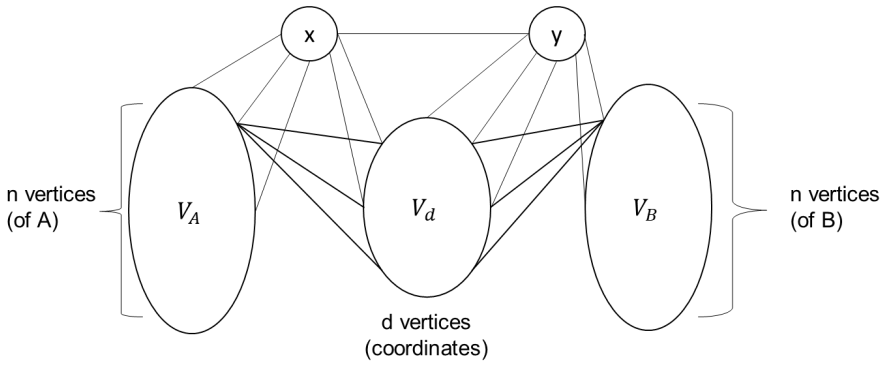

Figure 3: The Updated Graph G

- For all pairs not of the form $(a, b) \in V_A \times V_B$, distance is at most 2.
- For pairs $(a, b) \in V_A \times V_B$, we did not create a path of length at most 2 unless such a path already existed.

Conclusion

$$Diam\,(G) > 2 \Leftrightarrow \exists a \in V_A, b \in V_B,\ d_G\,(a, b) > 2 \Leftrightarrow \langle a, b \rangle = 0$$

Therefore, solving the diameter problem solves OV.
The graph size is: $|V| = 2n + d + 2 = O\,(n)$, $|E| = O\,(nd) = n^{1+o(1)}$

□

References

[1] Stephen A. Cook. "The complexity of theorem-proving procedures". In: *Proceedings of the Third Annual ACM Symposium on Theory of Computing*. STOC '71. Shaker Heights, Ohio, USA: Association for Computing Machinery, 1971, pp. 151–158. ISBN: 9781450374644. DOI: 10.1145/800157.805047. URL: https://doi.org/10.1145/800157.805047.

[2] Thomas Dueholm Hansen et al. "Faster k-sat algorithms using biased-ppsz". In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. 2019, pp. 578–589.

[3] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. "Which problems have strongly exponential complexity?" In: *Journal of Computer and System Sciences* 63.4 (2001), pp. 512–530.

[4] Richard M Karp. "Reducibility among combinatorial problems". In: *50 Years of Integer Programming 1958-2008: from the Early Years to the State-of-the-Art*. Springer, 2009, pp. 219–241.

[5] B. Monien and E. Speckenmeyer. "Solving satisfiability in less than 2n steps". In: *Discrete Applied Mathematics* 10.3 (1985), pp. 287–295. ISSN: 0166-218X. DOI: https://doi.org/10.1016/0166-218X(85)90050-2. URL: https://www.sciencedirect.com/science/article/pii/0166218X85900502.