CSS.414.1: Polynomial Methods in Combinatorics

Instructor: Mrinal Kumar

TIFR 2024, Aug-Dec

Scribe: Soham Chatterjee

sohamchatterjee999@gmail.com
Website: sohamch08.github.io

Contents

1 Introduction and Targets

The content of this course will be the followings:

- Polynomial Methods in Combinatorics/Geometry
  1. Kakeya/Nikodym Problem over finite fields
  2. Joints Problem
  3. Combinatorial Nullstellensatz (CN)
  4. CN proof of Cauchy-Devenport, Erdös-Heilbronn Conjecture
- Polynomial Methods in Algebraic Algorithms
  1. Noisy Polynomial Interpolation (Sudan, Guruswami-Sudan)
  2. Multiplicative noise (Von zur Gathen-Shparlinski)
  3. Coppersmith's Problem (Given an univariate $f(x) \in \mathbb{Z}[x]$, compute all 'small' integer roots modulo a composite)
- Polynomial Methods in Circuit Complexity
  1. Razborov-Smolensky (Lower Bound for constant depth AND, OR, NOT, $\bmod p$ gates)
  2. Algorithmic consequences (all pairs shortest paths)
  3. Upper bounds on matrix rigidity (Alman-Williams '2015, Dvir-Edelman '2017)
- Polynomial in Property Testing: Polischuk-Speilman Lemma/Variants
- Weil Bounds (Stepanov, Schmidtm Bombieri)
- Rational Approximations of Algebraic Numbers (Thue[1907] - Siegel - Roth[1954])

2 Joints Problem

3 Combinatorial Nullstellensatz

3.1 Chevally-Warning Theorem

4 Sum Sets

4.1 Sum Sets over Finite Fields

4.1.1 Cauchy-Davenport Theorem

4.2 Restricted Sum Sets

4.2.1 Erdös-Heilbronn Conjecture

5 Arithmetic Progression Free Sets in $\mathbb{F}_3^n$

5.1 3AP Free sets in $\mathbb{F}_q$

6 3-Tensors and Slice Rank

6.1 Rank

6.2 Generalization to 3-Dimension

6.3 Slice Rank of Diagonal 3D Tensor

7 Kakeya and Nikodym Problem

Definition 7.0.1: Kakeya Sets

In a finite field $\mathbb{F}_q$, $K \subseteq \mathbb{F}^n$ is a Kakeya Set if $\forall\, a \in \mathbb{F}^n$, $\exists\, b \in \mathbb{F}^n$ such that

$$L_{a,b} = \{b + at : t \in \mathbb{F}_q\} \subseteq K$$

i.e. informally it has a line in every direction

Now notice that we can take the whole $\mathbb{F}_q^n$ as the Kakeya Set. We can also remove a point from $\mathbb{F}_q^n$ and it will still be a Kakeya Set. Having defined the Kakeya sets the biggest question which is studied is:

Question 7.1

How small can a Kakeya Set be?

7.1 Lower Bound on Nikodym Sets

7.2 Lower Bound on Kakeya Sets

7.2.1 Hasse Derivative

8 Razborov Smolensky Lower Bound

The result we will discuss the result that majority is strictly harder than the parity for AC^0, since there is no polynomial-size AC^0 circuit to compute majority even if we are given parity gates. The result is Razborov's, and the proof technique uses ideas due to both Razborov and Smolensky.

Consider the class AC^0 of polynomial size circuits with constant depth with unbounded fan-in. We consider the class $\mathrm{AC}^0(\oplus)$ where we are give the parity gates $\oplus$ which outputs 1 if an odd number of its inputs are 1. The main theorem which we will prove in this section is:

Theorem 8.1 Razborov-Smolensky

For any $d \in \mathbb{N}$ any any depth d $\mathrm{AC}^0(\oplus)$ circuit for MAJORITY has size $\geq 2^{\Omega\left(n^{\frac{1}{2d}}\right)}$

8.1 Two Parts of Proving Lower Bound

The proof of the above theorem requires two lemmas:

Lemma 8.1.1

$\forall\ \epsilon > 0$ and $d \in \mathbb{N}$ the following is true:

If $f : \{0,1\}^n \longrightarrow \{0,1\}$ can be computed by a size s depth d $\mathrm{AC}^0(\oplus)$ circuit then $\exists$ a polynomial g in n variables and deg $O\left(\log \frac{s}{\epsilon}\right)^d$ such that

$$\mathop{\mathbb{P}}_{a\in\{0,1\}^n}[f(a) = g(a)] \geq 1 - \epsilon$$

Lemma 8.1.2

For all polynomials $p(x_1, \ldots, x_n)$ with $\deg p = t$,

$$\mathop{\mathbb{P}}_{a\in\{0,1\}^n}[g(a) = \textsc{Maj}(a)] \leq \frac{1}{2} + O\left(\frac{t}{\sqrt{n}}\right)$$

Now first we will show that with these two lemmas we can prove Razborov-Smolensky Lower Bound for MAJORITY function.

Proof of Theorem 8.1: Suppose MAJ has a $\mathrm{AC}^0(\oplus)$ circuit of size $< 2^{n^{\frac{1}{2d}-\delta}}$

$\xRightarrow{\text{Lemma 8.1.1}}$ $\exists$ polynomial g of degree $n^{\frac{1}{2d}-\delta}$ that approximates MAJ with error 0.1.

$\xRightarrow{\text{Lemma 8.1.2}}$ $\forall$ polynomial g of deg $n^{\frac{1}{2d}-\delta}$ the error is $\geq 1 - \left[\frac{1}{2} + O\left(\frac{n^{\frac{1}{2d}-\delta}}{\sqrt{n}}\right)\right] \geq \frac{1}{2} - - \left[\frac{1}{2} + O\left(\frac{n^{\frac{1}{2d}-\delta}}{\sqrt{n}}\right)\right] \geq \frac{1}{2} - o(1)$

But $\frac{1}{2} - o(1) < 0.1$ is contradiction. ■

Alternate Proof Theorem 8.1: Suppose C be an $\mathrm{AC}^0(\oplus)$ circuit of size s and depth d computing MAJORITY

$\xRightarrow{\text{Lemma 8.1.1}}$ $\exists$ polynomial g of degree $O\left(\log \frac{s}{\epsilon}\right)^d$ with error probability $\leq \epsilon$.

$\xRightarrow{\text{Lemma 8.1.2}}$ $\forall$ polynomial g of deg $O\left(\log \frac{s}{\epsilon}\right)^d$ the error is $\geq \frac{1}{2} + O\left(\frac{\left(\log \frac{s}{\epsilon}\right)^d}{\sqrt{n}}\right)$.

Hence from these two results and setting $\epsilon = 0.1$ we have

$$\frac{1}{2} + O\left(\frac{\left(\log \frac{s}{\epsilon}\right)^d}{\sqrt{n}}\right) \geq 1 - \epsilon \implies (\log 10s)^d \geq \sqrt{n} \implies s \geq 2^{\Omega\left(\frac{1}{2d}\right)}$$

■

Now that we proved our main objective theorem we will focus on proving the 2 lemmas in the following two sections.

8.2 Approximating Boolean Function with Polynomials

We first state and prove a lemma showing that every $\mathrm{AC}^0(\oplus)$ circuit can be approximated by a low degree polynomial i.e. Lemma 8.1.1. But to prove that we will show a more stronger lemma and then the lemma follows as a simple corollary of this stronger result.

Lemma 8.2.1

For all $\mathrm{AC}^0(\oplus)$ circuits C of size s of depth d and $\forall\, \epsilon > 0$ there exists a distribution $\mathscr{D}$ of polynomials $p(x_1,\dots,x_n) \in \mathbb{F}_2[x_1,\dots,x_n]$ such that for all $a \in \{0,1\}^n$

$$\mathop{\mathbb{P}}_{p\in\mathscr{D}}[p(a) = C(a)] \geq 1-\epsilon$$

where $\mathscr{D}$ is supported on polynomials of degree $\leq \left(\log \frac{s}{\epsilon}\right)^d$

First we will show that this lemma implies Lemma 8.1.1.

Proof of Lemma 8.1.1: Consider the $|\{0,1\}^n| \times |\operatorname{supp}\mathscr{D}|$ table for each $a \in \{0,1\}^n$, a represents a row in the table. In the table at $(a,i)^{th}$ entry put 1 if i^{th} polynomial p in $\mathscr{D}$ satisfies $p(a) = C(a)$. For rest of the positions put 0.

$\xRightarrow{\text{Lemma 8.2.1}}$ $\forall\, \epsilon > 0$ there exists a distribution $\mathscr{D}$ such that for all $a \in \{0,1\}^n$ such that $\mathop{\mathbb{P}}_{p\in(\mathscr{D})}[p(a) = C(a)] \geq 1-\epsilon$. Hence in the table for each $a \in \{0,1\}^n$, at least $1-\epsilon$ many fraction of $|\operatorname{supp}(\mathscr{D})|$ entries in a^{th} row have 1. Therefore there are total at least $(1-\epsilon)\cdot|\{0,1\}^n|\cdot|\operatorname{supp}(\mathscr{D})|$ many 1's in total in the table.

Hence by pigeon hole principle there is at least one column which has at least $(1-\epsilon)\cdot|\{0,1\}^n|$ many 1's. Therefore there is a polynomial $p \in \operatorname{supp}(\mathscr{D})$ which agrees with C in at least $1-\epsilon$ fraction of total inputs. Hence

$$\mathop{\mathbb{P}}_{a\in\{0,1\}^n}[p(a) = C(a)] \geq 1-\epsilon$$

■

Now we will prove the Lemma 8.2.1. Now before diving into the proof first let's see how can we approximate the gates in $\mathrm{AC}^0(\oplus)$ circuits with low-degree polynomials. That way we can approximate any $\mathrm{AC}^0(\oplus)$ circuit with low-degree polynomial.

So to for a $\neg x_i$ gate we can have the polynomial $1 - x_i$. For a $\bigoplus\limits_{i=1}^{k} x_i$ we can use the polynomial $\sum\limits_{i=1}^{k} x_i$. So only $\wedge$ and $\vee$ gates are remaining. Now notice if we have a low degree polynomial for $\wedge$ we also have a low degree polynomial for $\vee$ since

$$\bigvee_{i=1}^{n} x_i = \neg\left(\bigwedge_{i=1}^{n}(\neg x_i)\right)$$

So we will try to find a polynomial approximating an $\wedge$ gate of degree $\leq \left(\log \frac{1}{\epsilon}\right)^d$. We can't approximate $\wedge$ by outputting 0 every time since the desired correctness probability must hold for all inputs x. Multiplying a random constant-size subset of the bits will not work either, for the same reason.

Naive way to have a polynomial for $\bigvee\limits_{i=1}^{n} x_i$ would be $1 - \prod\limits_{i=1}^{n}(1-x_i)$. But with this the degree becomes very large.

Idea. *Check parity of random subset of $[n]$. So we take a random subset $S \subseteq [n]$ then we take the polynomial $p_S = \sum\limits_{i\in S} x_i$.*

Lemma 8.2.2

If S is a random subset of $[n]$ then

$$\mathop{\mathbb{P}}_{S\subseteq[n]}\left[p_S(x_1,\dots,x_n) = \bigvee_{i=1}^{n} x_i\right] \geq \frac{1}{2}$$

Proof: If $\overline{x} = (0,\dots,0)$ then we have $p_S(x_1,\dots,x_n) = \bigvee\limits_{i=1}^{n} x_i$. Suppose $\overline{x} \neq (0,\dots,0)$. Then only way $p_S(x_1,\dots,x_n) \neq \bigvee\limits_{i=1}^{n}$ is when S has an even number of 1 bits. So let $T \subseteq [n]$ such that $i \in T \iff x_i = 1$. Then $p_S(\overline{x}) = 0 \iff |S \cap T| \equiv 0 \bmod 2$. Now $|S\cap T| \bmod 2$ can be either 1 or 0. Since S is picked uniform at random the probability therefore the probability that $|S \cap T| \bmod 2 = 0$ is $\frac{1}{2}$. Therefore $\mathop{\mathbb{P}}_{S\subseteq[n], S\neq\emptyset}\left[p_S(x_1,\dots,x_n) \neq \bigvee\limits_{i=1}^{n} x_i\right] \leq \frac{1}{2}$. Hence we have $\mathop{\mathbb{P}}_{S\subseteq[n]}\left[p_S(x_1,\dots,x_n) \neq \bigvee\limits_{i=1}^{n} x_i\right] \geq \frac{1}{2}$ ■

Hence we if we pick a subset $S \subseteq [n]$ uniformly at random then with probability $\geq \frac{1}{2}$ we can approximate an $\vee$ gate or an $\wedge$ gate with a polynomial of degree 1. To have error $\frac{1}{2^k}$ we can chose k subsets of $[n]$ uniformly at random $S_1, \ldots, S_k$. Then construct the polynomial

$$p_{S_1,\ldots,S_k}(x_1, \ldots, x_n) = 1 - \prod_{i=1}^{k}\left(1 - p_{S_i}\right) = 1 - \prod_{i=1}^{k}\left(1 - \sum_{j \in S_i} x_j\right)$$

This has error probability $\frac{1}{2^k}$. So we can approximate $\vee$ gate or $\wedge$ gate with $\frac{1}{2^k}$ error probability with a degree k polynomial.

Proof of Lemma 8.2.1: So like the above discussion we replace each gate with polynomials starting with leaf and then we proceed to the top:

- For $\neg x_i$ gate replace by $1 - x_i$
- For $\bigoplus\limits_{i=1}^{n} x_i$ gate replace by $\sum\limits_{i=1}^{n} x_i$
- For $\bigvee\limits_{i=1}^{n} x_i$ gate uniformly pick k subsets $S_1, \ldots, S_k$ of $[n]$ then construct the polynomial

  $$p_{\vee}(x_1, \ldots, x_n) = 1 - \prod_{i=1}^{k}\left(1 - \sum_{j \in S_i} x_j\right)$$

  then the error probability becomes $\frac{1}{2^k}$ by Lemma 8.2.2. For $\bigwedge\limits_{i=1}^{n} x_i$ use the formula $\bigwedge\limits_{i=1}^{n} x_i = \neg\left(\bigvee\limits_{i=1}^{n}(\neg x_i)\right)$ use the process for $\vee$ gates. So

  $$p_{\wedge}(x_1, \ldots, x_n) = \prod_{i=1}^{n}\left(1 - \sum_{j \in S_i}(1 - x_j)\right)$$

  Here will choose k later so that we have the necessary total error.

The total polynomial for the circuit is constructed by composing of polynomials with each gate's S_j for $j \in [k]$ sampled from the input wires.

Now degree increases by a factor of k for each $\wedge$ gate or $\vee$ gate. Since the circuit has depth d, there can be $\vee$ gates or $\wedge$ gates in at most all depths. Hence degree of the final polynomial becomes $O(k^d)$.

For the error let ϵ_l denote the errors for each gate at depth l. Then for each gate g at depth $l-1$ we have error for g is $\leq \frac{1}{2^k} + |fanin(g)|\epsilon_l$.

Claim: $\epsilon_d \leq \frac{s}{2^k}$
Proof: We will prove this by induction. For base case $d = 1$ this is trivial. Let this is true for $d - 1$. For d consider all the children of the root gate v. Then

$$\epsilon_d \leq \frac{1}{2^k} + \sum_{u \in \text{Child}(v)} \frac{|C_u|}{2^k} = \frac{1 + \sum\limits_{u \in \text{Child}(v)} |C_u|}{2^k} = \frac{|C_v|}{2^k}$$

Hence by mathematical induction we have $\epsilon_d \leq \frac{s}{d}$ ■

Hence the total error is $\frac{s}{2^k}$. We want the error to be at most ϵ. Therefore

$$\frac{s}{2^k} \leq \epsilon \implies k = \log\frac{s}{\epsilon}$$

Hence the degree of the final polynomial approximating the circuit is $\left(\log\frac{s}{\epsilon}\right)^d$. Therefore the support of $\mathscr{D}$ has the polynomials of degree $\leq \left(\log\frac{s}{\epsilon}\right)^d$ ■

8.3 Degree-Error Trade of to Approximate Majority

Now we will prove the Lemma 8.1.2. But before that we first make some observations.

Note:-

The polynomial which approximates Majority can be made multilinear without changing its evaluation in $\{0,1\}^n$ just by replacing x_i^k by x_i for each variable and for each power.

Now we will show that if Maj has an approximating polynomial of low-degree then every $n-$variable boolean function $f:\{0,1\}^n \to \{0,1\}$ has an approximating polynomial of low degree.

Theorem 8.3.1 Versatility of Majority

$\forall\, f:\{0,1\}^n \to \{0,1\}, \exists\, g,h \in \mathbb{F}_2[x_1,\dots,x_n]$ such that

$$\forall\, x, f(x) = g(x)\cdot \text{Maj}(x) + h(x),\ \text{where}\ \deg g, \deg h \le \frac{n}{2}$$

Before proving this theorem first let's see what results we get from this theorem.

Lemma 8.3.2

Let $f \in \mathbb{F}_2[x_1,\dots,x_n]$ such that for all $x\in\{0,1\}^n$, $f(x)=\text{Maj}(x)$. Then $\deg f \ge \frac{n}{2}$.

Proof: Suppose $\exists\, p \in \mathbb{F}_2[x_1,\dots,x_n]$ such that $\deg p < \frac{n}{2}$ and for all $x\in\{0,1\}^n$ we have $p(x)=\text{Maj}(x)$.

$\xRightarrow{\text{Lemma 8.3.1}}$ $\forall\, f:\{0,1\}^n \to \{0,1\}$ such that $f(x) = g(x)\cdot\text{Maj}(x)+g(x)$ for all $x\in\{0,1\}^n$. Then the polynomial $f(x) = g(x)\cdot p(x) + h(x)$ for all $x\in\{0,1\}^n$. Then $\deg f \le n-1$. Hence all boolean function of $n-$variables can be computed by a polynomial of degree $\le n-1$.

But number of boolean functions over $n-$variables are 2^{2^n}. Number of polynomials of $n-$variables of degree $< n$ is $\le 2^{2^n}-1$. Hence there exists a boolean function which can not be computed by polynomial of degree $< n$. Contradiction. ■

Therefore $\deg(\text{Maj}) \ge \frac{n}{2}$. Now we will prove Lemma 8.1.2 using the above theorem.

Proof of Lemma 8.1.2: Let $p\in\mathbb{F}_2[x_1,\dots,x_n]$ be a polynomial of degree t. Let $S\subseteq\{0,1\}^n$ be the set of inputs where p and Maj agree.

$\xRightarrow{\text{Lemma 8.3.1}}$ $\forall\, f:\{0,1\}^n\to\{0,1\}$ there exists $g,h\in\mathbb{F}_2[x_1,\dots,x_n]$ with $\deg g, \deg g \le \frac{n}{2}$ such that $\forall\, z\in\{0,1\}^n$

$$f(a) = g(a)\text{Maj}(a) + h(a)$$

Hence every function $f|_S : S \to \{0,1\}^n$ can be computed by the polynomial $g(x)\cdot p(x)+h(x)\in\mathbb{F}_2[x_1,\dots,x_n]$ which has degree $\le \frac{n}{2}+t$.

Let $\mathcal{F}$ be the vector space of all functions $f|_S : S\to\{0,1\}$ for all $f:\{0,1\}^n\to\{0,1\}$ and let $\mathcal{P}$ be the vector space of all polynomials in $\mathbb{F}_2[x_1,\dots,x_n]$ of degree at most $\frac{n}{2}+t$. By the above argument we get that $\forall f|_S\in\mathcal{F}, \exists\, p_f\in\mathcal{P}$ such that $f|_S$ is computed by $\mathcal{P}$. Hence $\dim\mathcal{F}\le\dim\mathcal{P}$. Now

$$\dim\mathcal{P} = \sum_{i=0}^{\frac{n}{2}+t}\binom{n}{i} = \sum_{i=0}^{\frac{n}{2}}\binom{n}{i} + \sum_{i=\frac{n}{2}+1}^{\frac{n}{2}+t}\binom{n}{i} = \frac{1}{2}\,2^n + \sum_{i=\frac{n}{2}+1}^{\frac{n}{2}+t}\binom{n}{i} \le 2^{n-1} + t\frac{2^n}{\sqrt{n}} = 2^n\left(\frac{1}{2}+\frac{t}{\sqrt{n}}\right)$$

Now $\dim\mathcal{F} = |S|$. Hence

$$|S| \le 2^n\left(\frac{1}{2}+\frac{t}{\sqrt{n}}\right) \implies \frac{|S|}{2^n} \le \frac{1}{2}+\frac{t}{\sqrt{n}}$$

Therefore for any polynomial $p\in\mathbb{F}_2[x_1,\dots,x_n]$ with degree t we have $\underset{a\in\{0,1\}^n}{\mathbb{P}}[p(a)=\text{Maj}(a)] \le \frac{1}{2}+O\left(\frac{t}{\sqrt{n}}\right)$. ■

Observation. *Now let for any $f : \{0,1\}^n \to \{0,1\}$, $S_0 = \text{MAJ}^{-1}(0)$ and $S_1 = \text{MAJ}^{-1}(1)$. Suppose we can compute the polynomials $u, v \in \mathbb{F}_2[x_1, \ldots, x_n]$ with $\deg u, \deg v \leq \frac{n}{2}$ such that u,f agree on S_0 and v, f agree on S_1 i.e. $f|_{S_0}$ can be computed by u and $f|_{S_1}$ can be computed by v. Then $\forall\, x \in \{0,1\}^n$ we have*

$$f(x) = u(x)(1 - \text{MAJ}(x)) + v(x)\text{MAJ}(x)$$

Hence by the observation we can conclude that computing the polynomial for f on S_0 or S_1 is enough. Now we will prove the Versatility of MAJORITY Theorem.

Proof of Theorem 8.3.1: So assume $S_0 = \text{MAJ}^{-1}(0)$ and $S_1 = \text{MAJ}^{-1}(1)$. We want to show that these are interpolating sets for polynomials of degree $\leq \frac{n}{2}$ i.e. $\deg f|_{S_0}, \deg f|_{S_1} \leq \frac{n}{2}$.

Now we will show the process to find $u \in \mathbb{F}_2[x_1, \ldots, x_n]$ with $\deg u \leq \frac{n}{2}$ where u agrees with f in S_0. We will follow the same process to find the polynomial $v \in \mathbb{F}_2[x_1, \ldots, x_n]$ with $\deg v \leq \frac{n}{2}$ where v agrees with f in S_1. Now for any $S \subseteq [n]$ we denote $x^S := \prod_{i \in S} x_i$. Then

$$u(\overline{x}) = \sum_{S \subseteq [n], |S| \leq \frac{n}{2}} c_S x^S \quad \forall\, S \subseteq [n], |S| \leq \frac{n}{2},\ c_S \in \{0,1\}$$

We have to compute the coefficients of u. Now $u(a) = f(a)$ for all $a \in S_0$. Therefore we have a system of linear equations.

We we take a matrix M with rows and columns indexed by by subsets $S \subseteq [n]$ where $|S| \leq \frac{n}{2}$ and they are ordered so that the size is non-decreasing and lexicographically and use this same ordering for both the rows and columns. Now for $S \subseteq [n]$, $|S| \leq \frac{n}{2}$ the S^{th} column indicates the monomial x^S and the S^{th} row indicates the binary number $a \in \{0,1\}^n$ where $a_i = 1 \iff i \in S$. Naturally for any $S, T \subseteq [n]$ with $|S|, |T| \leq \frac{n}{2}$ we have $M(S,T) = 1 \iff S \subseteq T$. We have the coefficient vector C indexed same as rows of M as the variable vector and we have the column vector F_0 of values of f at every point of S_0. Then we have the equation $MC = F_0$.

To have a solution to exists we need $\det M \neq 0$. We will show M is an lower triangular matrix with all diagonal entries is 1. This is true because $M(S,T) = 1 \iff S \subseteq T$ therefore in the diagonal all entries are 1 and up the triangle it has 0's.. So we have $\det M \neq 0$. And therefore there is an unique solution for u.

Similarly we find v. And then we have $f(a) = u(a)(1 - \cdot\text{MAJ}(a)) + v(a)\text{MAJ}(a) = (v - u)(a) \cdot \text{MAJ}(a) + u(a)$. This proves the theorem. ■

9 All Pairs Shortest Path

ALL PAIRS SHORTEST PATH (APSP)
Input: $G = (V, E)$ adjacency matrix, $V = [n]$. G has no negative cycle.
Question: For all $u, v \in V$ output the length of the shortest path from u to v in G.

A very well known algorithm for solving this is using dynamic programming. Suppose SHORTEST-PATH(i, j, k)= Length of shortest path i to j that only takes vertices in $[k]$. So we have the following relation

$$\text{SHORTEST-PATH}(i,j,k) = \min\{\text{SHORTEST-PATH}(i,j,k-1), \text{SHORTEST-PATH}(i,k,k-1) + \text{SHORTEST-PATH}(k,j,k-1)\}$$

The time complexity of the algorithm is $O(n^3)$.

Theorem 9.1 [Wil14]
There is a randomized algorithm for APAP with runtime $\frac{n^3}{2^{\varepsilon\sqrt{\log n}}}$

Before diving in to the randomized algorithm we will first look into another problem.

MIN PLUS PRODUCT OF MATRICES (MPP)
Input: $A, B \in \mathbb{R}^{n \times n}$
Question: $C = A \circledast B$ with $C(i,j) = \min_k\{A(i,k) + B(k,j)\}$

Naively it takes $O(n^3)$ time to find the MPP of two matrices and no better algorithm with lesser time is known.

Theorem 9.2

1. APSP is in time $T(n) \implies$ MPP is in time $O(T(n))$.
2. MPP is in time $T(n) \implies$ APSP is in time $O(T(n)\log n)$.

Proof:

1. Let we have the matrices $A, B \in \mathbb{R}^{n\times n}$. Then we create the following 3-partite weighted graph $G(U, V, W, E)$ where $U = V = W = [n]$. For any $u \in U, v \in V, w \in W$ we have $w(u, v) = A(u, v)$ and $w(v, k) = B(v, k)$.

   For this graph we have that for any $i, j \in [n]$ $A \circledast B(i, j)$ = Length of the shortest path from $i \in U$ to $j \in W$. So now the problem is reduced to finding APSP in the constructed graph and from that we can get the MPP of the two matrices. So if we can solve APSP in $T(n)$ time then we can also solve MPP in $O(T(n))$ time.

2. Let we have the adjacency matrix of a graph $G = (V, E)$ of n vertices with self loops . Then we calculate the matrix
$$\hat{A} = \underbrace{A \circledast A \circledast \cdots \circledast A}_{n \text{ times}}$$
   by MPP. To calculate $\hat{A}$ efficiently we do repeated squaring to to have only $O(\log n)$ many MPP operations. Each operation takes $T(n)$ time. Hence if MPP can be solved in $T(n)$ time then APSP can be solved in $O(T(n)\log n)$ time.

■

Hence it is equivalent to show that there is an efficient algorithm for MPP with which we can get an algorithm for APSP

Theorem 9.3 [Wil14]
There is a randomized algorithm for MPP with runtime $\frac{n^3}{2^{\varepsilon\sqrt{\log n}}}$

Proof: So we have $A, B \in \mathbb{R}^{n\times n}$. We will break the algorithm into many steps to make it easy to understand.

Step 1 Reduction to Many Instances of Rectangular MPP: We first pick a parameter s such that $s \mid n$. Let $g = \frac{n}{s}$. Then we break A into g many $n \times s$ blocks and we break B into g many $s \times n$ blocks i.e.

$$\left[\begin{array}{c|c|c|c} A_1 & A_2 & \cdots & A_g \end{array}\right] \circledast \left[\begin{array}{c} B_1 \\ \hline B_2 \\ \hline \vdots \\ \hline B_g \end{array}\right]$$

So now we define $C_i = A_i \circledast B_i$. Then we have for any $i, j \in [n]$, $C(i, j) = \min_{j\in[g]} C_j$. Therefore it suffices to compute each C_i in time $\tilde{O}(n^2)$ time. From now on for brevity we will think A to be a $n \times s$ matrix and B to be a $s \times n$ matrix and we want to calculate the $C = A \circledast B$.

Step 2 Witness Matrix: Suppose we have the matrix $W \in \mathbb{R}^{n\times n}$ such that for any $i, j \in [n]$ we have
$$W(i, j) = k \text{ such that } k \text{ minimizes } A(i, k) + B(k, j),\ k \in [s]$$
Now having W we can compute $C = A \circledast B$ in $O(n^2)$ time since for each $i, j \in [n]$ we get the index k from $W(i, j)$ and then compute $A(i, k) + B(k, j)$ and it is equal to $C(i, j)$.

Therefore it suffices to compute W in $O(n^2)$ time. Now A, B may not have a unique witness matrix. Because for example suppose for some $i, j \in [n]$ there exists two indices k, k' such that $A(i, k) + B(k, j) = A(i, k') + B(k', j)$ and they are minimum. Hence there might be instances where for multiple k the value of C is minimum at that point. So we need to make the witness matrix unique first.

Step 3 Making Witness Matrix Unique: We have A, B from Step 1. We create the following two new matrices A', B' where for any $i, j \in [n]$ we have
$$A'(i, j) = (n + 1)A(i, j) + j \qquad B'(i, j) = (n + 1)B(i, j) + j$$
We claim that the corresponding witness matrix of A', B' is uniquely defined.

Claim 9.4

Witness matrix W' of A', B' is witness matrix W of A, B and moreover each entry is uniquely defined.

Proof: Let for any $i, j \in [n]$, $W'(i,j) = k_1$ then we have

$$A'(i,k_1) + B'(k_1,j) = (n+1)(A(i,k_1) + B(k_1,j)) + k_1 + j$$

Suppose there exists another $k_2 \in [s]$ such that $A'(i,k_2) + B'(k_2,j) = A'(i,k_1) + B'(k_1,j)$. Now two cases arise.

If $k_1 \neq k_2$ then either $A(i,k_1) + B(k_1,j) \neq A(i,k_2) + B(k_2,j)$ or $A(i,k_1) + B(k_1,j) = A(i,k_2) + B(k_2,j)$. If $A(i,k_1) + B(k_1,j) \neq A(i,k_2) + B(k_2,j)$ then we have

$$(n+1)([A(i,k_1) + B(k_1,j)] - [A(i,k_2) + B(k_2,j)]) = k_2 - k_1 > 0$$

$k_2 - k_1$ can never make for the $(n+1)$ since $(n+1)[A(i,k_1) + B(k_1,j)]$ and $(n+1)[A(i,k_2) + B(k_2,j)]$ at least differ by a factor of $n+1$ and $k_2 - k_1 < n+1$. Else $k_1 \neq k_2 \implies A'(i,k_1) + B'(k_1,j) \neq A'(i,k_2) + B'(k_2,j)$.

Therefore the witness of A', B' is a witness of A, B and its entries are uniquely defined ■

Therefore with this we get an unique witness matrix W'. From now on for brevity we will call A' by A, B' by B and W' by W.

Step 4 Fredman's Trick: Now we have a witness matrix W such that $\forall\, i, j \in [n]$ then $W(i,j) =$ unique $k \in [s]$. Now $\forall\, l \in [s]$ we have

$$A(i,k) + B(k,j) \leq A(i,l) + B(l,j) \iff A(i,k) - A(i,l) \leq B(l,j) - B(k,j)$$

So we create the matrices

$$\tilde{A}_i(k,l) = A(i,k) - A(i,l) \qquad \tilde{B}_j(k,l) = B(l,j) - B(k,j)$$

For $i \in [n]$, $\tilde{A}_i$ and $\tilde{B}_i$ are $s \times s$ matrices.

■

10 Matrix Rigidity

Definition 10.1: Rigidity of a Matrix

The rigidity of a matrix $A \in \mathbb{F}^{n\times n}$ for rank r over $\mathbb{F}$ (denoted by $R_A^{\mathbb{D}}(r)$) is the minimum number of entries to be changed in A so that the rank becomes at most r. More formally

$$R_A^{\mathbb{F}}(r) \triangleq \min_C \{sparsity(C) \mid C \in \mathbb{F}^{n\times n}, \operatorname{rank}(A+C) \leq r\}$$

Definition 10.2: (r,s) Rigid Matrix

A matrix $A \in \mathbb{F}^{n\times n}$ is said to be (r,s) rigid if it cannot be written as a sum of rank r matrix and a s-sparse matrix i.e. $R_A^{\mathbb{F}}(r) > s$.

11 References

[Wil14] Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, volume 12 of *STOC '14*, pages 664–673. ACM, May 2014.