Michael A. Forbes
mforbes@illinois.edu

9 -> 10

written 2

cs598maf Algebraic and Geometric Complexity Theory: Lecture 1 (2023-01-17)

logistics: - signup - piazza
- gradescope

today: - introduction
- bipartite matching

---

lecture: TR 14-15:15, DCL 1310 [here]

staff: instructor: Prof Michael A. Forbes (mforbes) [netid] T3:30 Siebel 3220 [office hrs] [after class, move to my office]

resources: website: courses.engr.illinois.edu/cs598maf/sp2023 [full policy details]

calendar: - lectures - topics
- materials - boardnotes [these notes]
- videos
- readings [supplement], non-replacement for lecture

forum (piazza): - course annc [sign up!]
- peer discussion

submission (gradescope): - coursework submission/return [signup!]
- gradebook

grades: 1 pset every 2 weeks

[informal]

prereq: ugrad algebra [duh]
complexity theory [duh]
randomized algo [less critical]

maturity

[main question]

Q: power of algebra in computation? [meaning?]

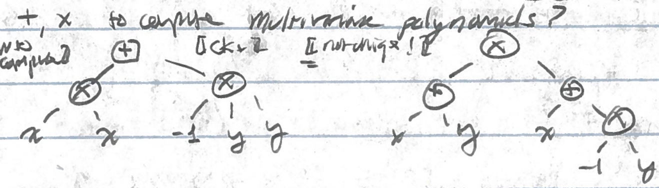
[many meanings] - graph theory
- linear algebra
- rings, fields, etc [etc]

[many mean] - algebraic computation [alg methods for alg objects] [diam of rubik's cube graph] [rank of matrix]
- boolean computation [Turing machine] [string equality] [not algebraic] [combinatorial optimization] [today]

[this course]

Q: power of +, × to compute multivariate polynomials?

$x^2 - y^2$ = [how to compute] [ckt] [non-unique!] [power of this sort of diagram?]



Q: fast algebraic algorithms? (upper bounds)
- nontrivial polynomial identities, eg det(XY) = det(X) det(Y) [normal form] [is an algo!]
- structural results, eg [every] poly can be expressed "as a determinant"
- efficient algebraic computation, eg fast matrix multiplication

Q: limits of algebraic algorithms? (lower bounds)
- algebraic P vs NP? [P vs NP]
- [exist] polynomials hard to compute? [most are!]
- [explicit] ________ [more interesting]

Q: role of randomness in algebraic algorithms?
- algebraic P vs BPP? [P vs BPP]
- can algebraic algorithms be efficiently derandomized? [randomness is pervasive in algebra]

BSS model

- deterministically and efficiently test if a polynomial is identically zero? (polynomial identity testing)
  - $\det(XY) = \det(X)\det(Y)$ [imagine how to prove this] [this is the main question]
  - $x^2 - y^2 = \triangle - \triangle$

Q: use algebraic geometry to understand algebraic algorithms?
↳ study of solution sets to systems of multivariate polynomials
- counting over infinite fields, via dimension [hard!] [existence of hard polynomials?]
- topological/Zariski closure of efficient algebraic computation [makes math nicer] [changes notion of "computation", redo thy] [VNP]

= [Q]

Q: why are we in this huge room?

[courses]	[edges]	[room/time slots]
cs 598 maf		Siebel 1302
cs 598 jge [Jeff]		1107 [for]
cs 598 rm [Ruta]		1105
		1103
	[match]	⋮
		DCL 1310 [us]

} consumed by students

def: an undirected graph $G=(V,E)$ is
- set of vertices V
- set of unordered pairs $E \subseteq \binom{V}{2}$

is bipartite if $V = L \cup R$ [disjoint], $E \subseteq L \times R$ (eg)

def: G undirected. A matching M is $M \subseteq E$ w/ each $v \in V$ incident to ≤ 1 edge
is perfect ——— $= 1$ edge

the bipartite perfect matching problem is to decide if a given bipartite graph has a perfect matching [and possibly also find it]

thm [Ford Fulkerson 54] [prehistory]: ↳ solvable in $O(nm)$ time ($n = |V|$, $m = |E|$)

sketch [cs473]: algo: start w/ empty matching
while "augmenting path" exists
augment matching w/ augmenting path
[decision] accept iff final matching is perfect

correctness: augmenting path exists iff matching not maximum ← max-flow = min-cut

complexity: $O(n)$ loop iterations
- matching size increases w/ each augment
- matching size $\leq$ # vertices [$= n$]

$O(m)$ work per iteration [find augmenting path] [breadth first search] □

rmk: algorithm is inherently sequential [n loop iterations]

[later...] thm [Hopcroft Karp 73] [start of TCS?] ——— in $O(\sqrt{n} \cdot m)$ time

idea: augment many paths at once, in $O(\sqrt{n})$ iterations

[lots of related work, no headlines]
[combinatorial algo]

Thm [Madry 13] : —— in $\tilde{O}(m^{10/7})$ time

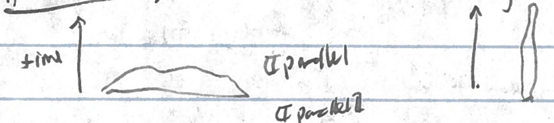
thm [CKLPG 22] : —— in $m^{1+o(1)}$ time [nearly linear !]

→ many sophisticated ideas
[optimization]
- solve continuous version via interior point methods
- dynamic data structures
- ...

[?] — rmk: these algorithms take $n^{\Omega(1)}$ [sequential] time

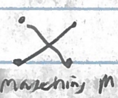
Q [parallel] algo for bipartite matching? doubling computer power halves the runtime?

eg: [sequential] [work is same, time is not]



eg: voting recounts, vs driving to Chicago

idea: turn bipartite perfect matching into algebraic problem.



$\simeq$ f_M [poly associated to M]

{perfect matching M in G} $\simeq$ $\sum_M f_M = f_G$

" ⇒ " perf match in G iff $f_G \neq 0$
↳ solve via polynomial identity testing

def: X n×n symbolic matrix $(X)_{ij} = x_{ij}$ [distinct vars]

the permanent $\text{perm}(X) = \sum_{\substack{\sigma:[n]\to[n] \\ \sigma \text{ permutation}}} x_{1\sigma(1)} x_{2\sigma(2)} \cdots x_{n\sigma(n)}$

lem: $G = ([n] \sqcup [n], E)$ bipartite, with $L = [n]$, $R = [n]$, define X_G by $(X_G)_{ij} = \begin{cases} x_{ij} & (i,j) \in E \\ 0 & \text{else} \end{cases}$ [symbolic adjacency matrix]

$\text{perm}(X_G) \neq 0$ iff G has a perfect matching

pf. $\text{perm}(X_G) = \sum_{\substack{\sigma:[n]\to[n] \\ \text{perm}}} \prod_{i=1}^{n} (X_G)_{i\sigma(i)}$ ← = 0 if edge not present

$= \sum_{\sigma:[n]\to[n]} \prod_i x_{i\sigma(i)}$, with $L = [n]$, $R = [n]$

[σ permutation using edges of G] ≡ σ is perfect matching

$= \sum_{\text{perf matching } M \text{ in } G} f_M(\bar{x})$ ← distinct monomials, so no cancellation

$\neq 0$ iff perfect matching exists

[does this help?]

def: the polynomial identity testing problem (PIT) is to,
when given a polynomial, determine whether it is zero [an identity]
vague [find on side? / given formally?]

lem [Schwartz Zippel]: $f \in \mathbb{F}[x_1, \ldots, x_n]$, $S \subseteq \mathbb{F}$. $\Pr_{\bar{\alpha} \leftarrow S^n}[f(\bar{\alpha}) = 0] \leq \frac{\deg f}{|S|}$

idea: algo:
- $\bar{\alpha} \leftarrow S^n$
- accept iff $f(\bar{\alpha}) = 0$

correctness: "clear" [unlike hardness] $f = 0 \Rightarrow \forall \bar{\alpha}\ f(\bar{\alpha}) = 0$
$f \neq 0 \Rightarrow$ wgp $f(\bar{\alpha}) \neq 0$

complexity: 1 evaluation to f

Q: efficient (algebraic) algorithm for permanent?

thm [Valiant]: permanent is #P-hard [unlikely to be efficient] (counting)

goal [main]: [unconditionally] prove the permanent is hard to compute algebraically [can do in special cases]

Q: circumvent this

def: the determinant $\det(X) = \sum_{\sigma} \operatorname{sgn}(\sigma) \prod_i x_{i\sigma(i)}$ [same sum] ($\operatorname{sgn}(\sigma) \in \{\pm 1\}$)

lem: G has perf match iff $\det(X_G) \neq 0$

cor [via PIT]: randomized algo for bipartite perfect match

sketch:
algo: ____
correctness: ____
complexity: - det in $O(n^{\omega + \epsilon})$ time $\forall \epsilon > 0$ (ω: matrix mult exponent < 2.373)
- $O(\lg^2 n)$ [parallel] time w/ poly(n) work

□

rmk: - $n^{\omega} < \sqrt{n}\, m$ [HK73] if $m \gg n$ "dense graphs" [so this was only recently superseded]
- [only] known parallel algo for bipartite perfect matching
- generalizes
  - [output?] the matching, in parallel [search vs decision]
  - [non] bipartite [max] matching, w/ (small) [weights] [nonnegative ...]
  - linear matroid intersection
  - "exact" matching [P?]

goal [main]: efficiently and [deterministically] solve PIT [can do in special cases]

"thm": our two goals are the same!

today: - introduction
- bipartite matching

next lecture: - computational models
- structural results

logistics: - signup - pizza
- gradescope