On The Complexity of Symmetric Polynomials

2025/07/08
{Unclear what this seminar is called}

(Markus Bläser (Saarland), Gorav Jindal (Aalto) ITCS 19)

Throughout, fix n and work in $\mathbb{C}[x_1, \ldots, x_n] = \mathbb{C}[\bar{x}]$

Thm [Waring 1770, Gauss 1876]:

Let $f_{sym}(x_1, \ldots, x_n) \in \mathbb{C}[x_1, \ldots, x_n]^{G_n}$. Then $\exists!\ f \in \mathbb{C}[x_1, \ldots, x_n]$ s.t. $f(e_1, \ldots, e_n) = f_{sym}(x_1, \ldots, x_n)$

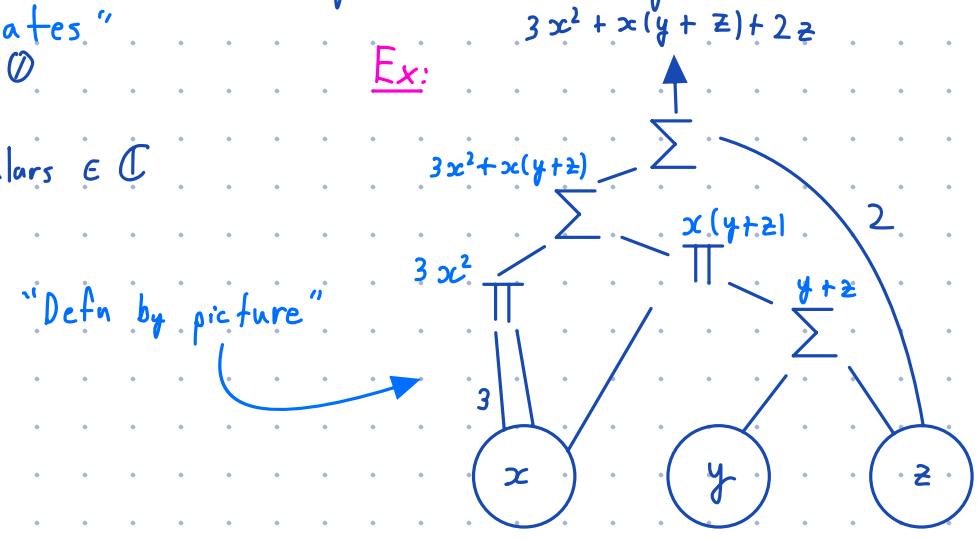
Pf: Sturmfels' "Algorithms in Invariant Theory" has a nice write up.

Q [Lipton-Regan 2009]: How do the "complexity" of f_{sym} and f relate?

Defn: An algebraic circuit over $\mathbb{C}[\bar{x}]$ is a directed acyclic graph w/:

- Nodes of in-degree 0 labelled w/ variables x_i or scalars $\in \mathbb{C}$ "inputs"
- Nodes labeled by Σ or Π w/ in-degree 2, out-degree unbounded "sum and product gates"
- Nodes of out-degree 0 "output (s)"
- Edges labelled by scalars $\in \mathbb{C}$ "scalar multiplication"

Ex:



"Defn by picture"

The size of a circuit is given by the number of vertices
For $f \in \mathbb{C}[\bar{x}]$, $L(f)$ is the size of the smallest circuit computing f
We can have multiple outputs and can define $L(\{f_1, \ldots, f_k\})$

Defn: For $g \in \mathbb{C}[x_1, \ldots, x_n]$, we may want to consider additional "g-gates" with in-degree n and out-degree unbounded. These gates add size 1.
We will call circuits with such gates g-oracle circuits.

$L^g(f)$ is the size of the smallest g-oracle circuit computing f

Ex [Ben-Or's Trick (1999)]: $e_1, \dots, e_n$ are computed by depth-3 $\Sigma\Pi\Sigma$ circuits of size $O(n^2)$

Pf: Evaluate the following polynomial $F(y) \in \mathbb{C}[x_1, \dots, x_n][y]$ at $\alpha_1, \dots, \alpha_{n+1}$

$$F(y) = \prod_{i=1}^{n} y - x_i = \sum_{i=0}^{n} (-1)^i y^{n-i} e_i(x_1, \dots, x_n)$$

Let $\beta_i := f(\alpha_i)$

$$\begin{pmatrix} \alpha_1^n & \alpha_1^{n-1} & \cdots & \alpha_1 & 1 \\ \alpha_2^n & \alpha_2^{n-1} & \cdots & \alpha_2 & 1 \\ & & \vdots & & \\ \alpha_n^n & \alpha_n^{n-1} & \cdots & \alpha_n & 1 \\ \alpha_{n+1}^n & \alpha_{n+1}^{n-1} & \cdots & \alpha_{n+1} & 1 \end{pmatrix} \begin{pmatrix} e_0 \\ -e_1 \\ \vdots \\ e_{n-1} \\ (-1)^n e_n \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \\ \beta_{n+1} \end{pmatrix}$$

LHS is Vandermonde which is invertible

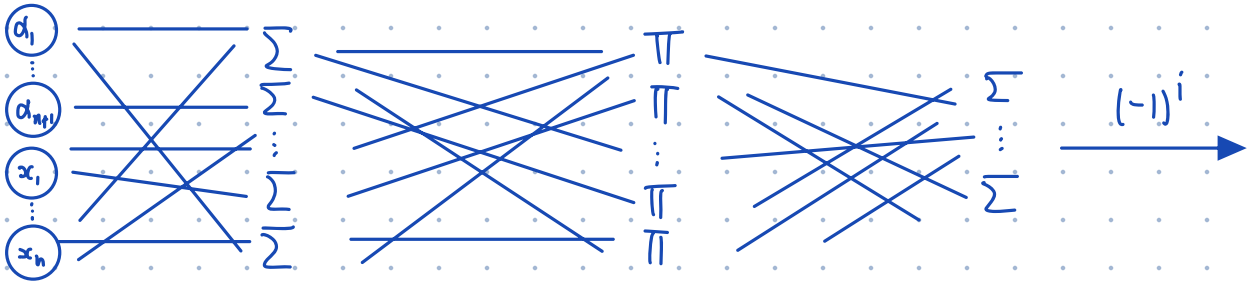
Denote the inverse by $(z_{i,j})_{1 \le i,j \le n+1}$

With some work, one can find an exact formula for $(z_{i,j})$ in terms of $(\alpha_i^{(n+1-j)})$

$$\begin{pmatrix} z_{11} & z_{12} & \cdots & z_{1n} & z_{1,n+1} \\ z_{21} & z_{22} & \cdots & z_{2n} & z_{2,n+1} \\ & & \vdots & & \\ z_{n1} & z_{n2} & \cdots & z_{nn} & z_{n,n+1} \\ z_{n+1,1} & z_{n+1,2} & \cdots & z_{n+1,n} & z_{n+1,n+1} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \\ \beta_{n+1} \end{pmatrix} = \begin{pmatrix} e_0 \\ -e_1 \\ \vdots \\ e_{n-1} \\ (-1)^n e_n \end{pmatrix}$$

one can move this sign inside to get $\Sigma\Pi\Sigma$

$$e_i(x_1, \dots, x_n) = (-1)^i \sum_{j=1}^{n+1} z_{i,j} \prod_{k=1}^{n} \alpha_j - x_k$$



$O(n^2)$ inputs — compute $\alpha_j - x_k$ $\Rightarrow O(n^2)$ Σ gates — product of n terms, do for each j $\Rightarrow O(n^2)$ Π gates — sum of $n+1$ terms $\Rightarrow O(n)$ Σ gates

1) We are not taking into account time to compute $z_{i,j}$'s. We just know they exist and so we can use them.

2) We cannot compute $F(\alpha_j) = \prod_{k=1}^{n} \alpha_j - x_k$ ahead of time as that is not an element of our base field $\mathbb{C}$

Rmk: Above $\Rightarrow L^f(f_{sym}) = O(n^3)$, can get down to $O(n^2)$ w/ cleverness

Thm [Bläser, Jindal]:

Let $f_{sym} = f(e_1, \dots, e_n) \in \mathbb{C}[x_1, \dots, x_n]^{\mathfrak{S}_n}$, $d = \deg(f)$, $d_{sym} = \deg(f_{sym})$

1) $L^{f_{sym}}(f) = \tilde{O}(n^3 \cdot d^2 \cdot d_{sym})$ $\quad \tilde{O}$ hides poly-log factors from big-O

2) $L(f) = \tilde{O}(d^2 \cdot L(f_{sym}) + d^2 n^2)$

Idea: Consider $n = 2$

$$B(y) = \prod_{i=1}^{2} y - x_i = y^2 - \overbrace{(x_1 + x_2)}^{e_1} y + \overbrace{x_1 x_2}^{e_2} \quad \text{vanishes at } x_1, x_2$$

$$\Rightarrow x_1 = \frac{e_1 + \sqrt{e_1^2 - 4e_2 - 1}}{2}, \quad x_2 = \frac{e_1 - \sqrt{e_1^2 - 4e_2 - 1}}{2}. \quad (*)$$

Let $f_{sym}(x_1, x_2) = f(e_1, e_2) \in \mathbb{C}[x_1, x_2]^{\mathfrak{S}_2}$, $d = \deg(f)$

Substituting $(*)$ for x_1, x_2 in $f_{sym}(x_1, x_2)$ yields $f(e_1, e_2)$.
However, cannot compute $\sqrt{e_1^2 - 4e_2}$ with algebraic circuits.

Lem: Let $F = y^n + f_1(u_1, \dots, u_n) y^{n-1} + \dots + f_n(u_1, \dots, u_n)$ monic square-free.
If $F(y, 0, \dots, 0)$ has n simple roots, the roots $A_1(u_1, \dots, u_n), \dots, A_n(u_1, \dots, u_n)$ of $F(y, u_1, \dots, u_n)$ can be expanded as power series.

Substitute $e_2 \leftarrow e_2 - 1$ in $(*)$
Then $\exists$ a power series of $\sqrt{e_1^2 - 4(e_2 - 1)}$ via Taylor series.
Since $\deg f = d$, we only need deg d truncations of these power series.
Then undo our substitution $e_2 \leftarrow e_2 + 1$.

More specifically, $F(y, u_1, \dots, u_n) = y^n - e_1 y^{n-1} + \dots + (-1)^n (e_n + (-1)^{n-1})$

then $F(y, 0, \dots, 0) = y^n - 1$ which has n simple roots.

So we need to compute roots of power series, then truncate terms.
(Newton Iteration) (Extraction)

Defn: $I = \langle u_1, \ldots, u_n \rangle$

Algorithm 2 Inverse computation.

Input: A circuit C computing the polynomial $g(u_1, u_2, \ldots, u_n)$ such that $g(0, 0, \ldots, 0) \neq 0$ and a positive integer d with $d = 2^\ell$ for some $\ell \in \mathbb{N}$.

Output: A circuit D for computing a polynomial $p(u_1, u_2, \ldots, u_n)$ such that $p \equiv g^{-1} \bmod I^d$, here $I = (u_1, u_2, \ldots, u_n)$ and g^{-1} is the inverse of g in $\mathbb{C}[[u_1, u_2, \ldots, u_n]]$.

1: $p_0 \leftarrow \frac{1}{g(0,0,\ldots,0)}$.
2: **for** $0 \leq k \leq \ell - 1$ **do**
3: $\quad p_{k+1} \leftarrow p_k \cdot (2 - g \cdot p_k)$.
4: **end for**
5: **return** p_ℓ.

Lem: Alg. 2 computes polynomials p_k s.t. $p_k = g^{-1} \pmod{I^{2^k}}$ for $0 \leq k \leq \ell$

Pf:
Induct on k. Trivial for $k = 0$, so sps $k > 0$.

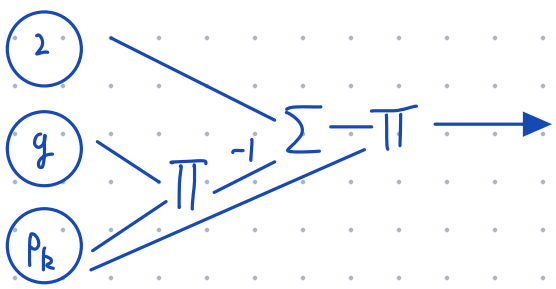
$$\frac{1}{g} - p_{k+1} = \frac{1}{g} - p_k \cdot (2 - g \cdot p_k)$$

$$= g \cdot \left(\frac{1}{g^2} - \frac{2p_k}{g} + p_k^2 \right) = g \cdot \left(\frac{1}{g} - p_k \right)^2$$

$$\frac{1}{g} - p_k \in I^{2^k} \Rightarrow \left(\frac{1}{g} - p_k \right)^2 \in I^{2^{k+1}}$$

Cor: Let g and $p = p_d$ as above. Then $L(p) \leq L(g) + O(\log d)$

Pf:




This is a common technique: rephrase algorithms in terms of a circuit

However, we can't do this for every algorithm.
ex: can't check for equality

Induction $\Rightarrow$ $L(p) \leq L(g) + 3 \cdot \log d = L(g) + O(\log d)$

Algorithm 1 Newton's Method.

Input: A square free monic polynomial $F(y) = F(y, u_1, u_2, \ldots, u_n) \in \mathbb{C}[u_1, u_2, \ldots, u_n][y]$ with respect to y of degree n such that $F(y, 0, 0, \ldots, 0)$ has n simple roots. A positive integer d with $d = 2^\ell$ for some $\ell \in \mathbb{N}$. We assume that $A_1, A_2, \ldots, A_n \in \mathbb{C}[[u_1, u_2, \ldots, u_n]]$ are the roots of $F(y)$.

Output: Degree d truncations $A_1^{(\ell)}, A_2^{(\ell)}, \ldots, A_n^{(\ell)}$ of the n roots $(A_1, A_2, \ldots, A_n)$ of $F(y)$, that is, $A_i^{(\ell)} \equiv A_i \bmod I^d$ with $I \stackrel{\text{def}}{=\!=} (u_1, u_2, \ldots, u_n)$ for all $i \in [n]$.

1: $\{\alpha_1, \alpha_2, \ldots, \alpha_n\} \leftarrow$ Roots of $F(y, 0, 0, \ldots, 0)$.
2: **for** $1 \le i \le n$ **do**
3: $\quad A_i^{(0)} \leftarrow \alpha_i$.
4: $\quad$ **for** $0 \le k \le \ell - 1$ **do**
5: $\quad\quad A_i^{(k+1)} \leftarrow A_i^{(k)} - \frac{F(A_i^{(k)})}{F'(A_i^{(k)})}$.
6: $\quad$ **end for**
7: **end for**
8: **return** $A_1^{(\ell)}, A_2^{(\ell)}, \ldots, A_n^{(\ell)}$.

Thm: For all $1 \le i \le n$, $A_i^{(k)} = A_i \pmod{I^{2^k}}$ for $0 \le k \le \ell$

Pf:
Induct on k. Trivial for $k = 0$, so sps $k > 0$.
Fix $1 \le i \le n$,

$$0 = F(A_i) = F\left(A_i^{(k)} + A_i - A_i^{(k)}\right)$$
$$= F(A_i^{(k)}) + (A_i - A_i^{(k)}) F'(A_i^{(k)}) + \sum_{j>1} \frac{(A_i - A_i^{(k)})^j}{j!} F^{(j)}(A_i^{(k)})$$

$F'(\alpha_i) =$ constant term of $F'(A_i^{(k)}) \neq 0$ since α_i is a simple root
$\Rightarrow F'(A_i^{(k)})$ invertible in $\mathbb{C}[[u_1, \ldots, u_n]]$

$$A_i - A_i^{(k+1)} = A_i - \left(A_i^{(k)} - \frac{F(A_i^{(k)})}{F'(A_i^{(k)})}\right) = -\sum_{j>1} \frac{(A_i - A_i^{(k)})^j}{j!\, F'(A_i^{(k)})} F^{(j)}(A_i^{(k)})$$

$A_i - A_i^{(k)} \in I^{2^k} \Rightarrow (A_i - A_i^{(k)})^j \in I^{2^{k+1}}$ for $j > 1$

Top of a board, don't erase — w/ defn of T

With all of the above, fix $F(y, e_1, \ldots, e_n) = y^n - e_1 y^{n-1} + \cdots + (-1)^n (e_n + (-1)^{n-1})$

The e_i are variables even though we are thinking of them eventually as elementary symmetric polynomials

I'll leave this computation as an exercise; it's a good check of how algebraic circuits behave with "reusing" computation

Lem: $F(y)$ and $F'(y) = \frac{\partial}{\partial y} F(y, e_1, \ldots e_n)$ have circuits of size $O(n)$.

Cor: Let $D_i := A_i^{(\ell)}$, $\ell = \log(d)$, for $1 \le i \le n$. Then $L(\{D_1, \dots, D_n\}) \le O(n^2 \ell + n\ell^2)$

$A_i^{(0)} = \alpha_i$ has a circuit of size 1.

$$A_i^{(k+1)} = A_i^{(k)} - \frac{F(A_i^{(k)})}{F'(A_i^{(k)})} \Rightarrow L(A_i^{(k+1)}) \le L(A_i^{(k)}) + O(n + \log d)$$

$$\Rightarrow L(D_i) \le O(n \log(d) + \log(d)^2)$$

Lem [Folklore]: Let $f(x_1, \dots, x_n) \in \mathbb{C}[x_1, \dots, x_n]$, $d = \deg(f)$.
Then $\forall\, 0 \le m \le d$, $L^f(f^{[m]}) = O(nd)$.

★ Leave these Lemmas on board ★

Pf: Let y a new indeterminate. Then

$$F(y) = f(yx_1, \dots, yx_n) = f^{[0]} + f^{[1]} y + \cdots + y^d f^{[d]}$$

Interpolate at $d+1$ points $\alpha_1, \dots, \alpha_{d+1}$.

The same Vandermonde argument $\Rightarrow \exists\, z_{i,j} \in \mathbb{C}$, $1 \le i, j \le d+1$ s.t.

$$f^{[m]} = (-1)^m \sum_{j=1}^{d+1} z_{m,j} \cdot F(\alpha_j)$$

Lem [Strassen 1973]: Let $f(x_1, \dots, x_n) \in \mathbb{C}[x_1, \dots, x_n]$, $d = \deg(f)$.
Then $\forall\, 0 \le m \le d$, $L(\{f^{[0]}, \dots, f^{[m]}\}) = O(m^2 \cdot L(f))$

Pf: ⟪ If time, gate simulation ⟫

We now have everything to prove the main theorem

Thm [Bläser, Jindal]:

Let $f_{sym} = f(e_1, \dots, e_n) \in \mathbb{C}[x_1, \dots, x_n]^{\mathfrak{S}_n}$, $d = \deg(f)$, $d_{sym} = \deg(f_{sym})$

1) $L^{f_{sym}}(f) = \tilde{O}(n^3 \cdot d^2 \cdot d_{sym})$

2) $L(f) = \tilde{O}(d^2 \cdot L(f_{sym}) + d^2 n^2)$

Pf: ★ $f_{sym}(\tilde{A}_1, \ldots, \tilde{A}_n) = f_{sym}(x_1, \ldots, x_n) = f(e_1, \ldots, e_n)$ ★

Let $C_{sym}(x_1, \ldots, x_n)$ be a circuit computing $f_{sym}(x_1, \ldots, x_n)$.

We cannot compute the roots $\tilde{A}_i$ of

$$\prod_{i=1}^{n} y - x_i = y^n - e_1 y^{n-1} + \cdots + (-1)^n e_n$$

and substitute them $x_i \leftarrow \tilde{A}_i$ into $C_{sym}(x_1, \ldots, x_n)$

Let $F(y, e_1, \ldots, e_n) = y^n - e_1 y^{n-1} + \cdots + (-1)^n (e_n + (-1)^{n-1})$
Again, let the roots of $F(y)$ be $A_1(e_1, \ldots, e_n), \ldots, A_n(e_1, \ldots, e_n)$

$\Rightarrow C_{sym}(A_1, \ldots, A_n) = f(e_1, \ldots, e_n + (-1)^{n-1})$

Then we saw we can compute polynomials $D_1, \ldots, D_n$ s.t. $A_i \equiv D_i \pmod{I^{d+1}}$ with a $O(n^2 \log d + n \log(d)^2)$ circuit

Let $h := C_{sym}(D_1, \ldots, D_n) = f(e_1, \ldots, e_n + (-1)^{n-1}) + g$, $g \in I^{d+1}$

$\Rightarrow L^{f_{sym}}(h) = O(n^2 \log(d) + n \log(d)^2)$.

We now split the proof into two parts

1) Deg $D_i \leq 2d \Rightarrow \deg(h) \leq 2 \cdot d \cdot d_{sym}$

$$\Rightarrow L^h(h^{[m]}) \leq O(n \cdot d \cdot d_{sym})$$

$$\sum_{m=0}^{d} h^{[m]} = f(e_1, \ldots, e_n + (-1)^{n-1}) \Rightarrow L^{f_{sym}}\left(\sum_{m=0}^{d} h^{[m]}\right) = \tilde{O}(n^3 \cdot d \cdot d_{sym})$$

Mapping $e_n \longmapsto e_n - (-1)^{n-1} \Rightarrow L^{f_{sym}}(f) = \tilde{O}(n^3 \cdot d \cdot d_{sym})$

2) $L(h) = O(L(f_{sym}) + n^2 \log(d) + n \log(d)^2)$

$\Rightarrow L(\{h^{[0]}, \ldots, h^{[d]}\}) = \tilde{O}(d^2 L(f_{sym}) + d^2 n^2)$

$\Rightarrow L(f) = \tilde{O}(d^2 L(f_{sym}) + d^2 n^2)$