Algebraic Complexity Theory

Lecture 3: Classes VP, VBP and VF

Department of Computer Science,
Indian Institute of Science

Recap

- In the last lecture, we saw that Det_n can be computed by a circuit of size $O(n^4)$ and depth $O((\log n)^2)$. We also saw that the *inverse* of an $n \times n$ matrix can be computed by a (multi-output) circuit of size $O(n^4)$ and depth $O((\log n)^2)$. Also, rank computation is in NC.

Recap

- In the last lecture, we saw that Det_n can be computed by a circuit of size $O(n^4)$ and depth $O((\log n)^2)$. We also saw that the *inverse* of an $n \times n$ matrix can be computed by a (multi-output) circuit of size $O(n^4)$ and depth $O((\log n)^2)$. Also, rank computation is in NC.

- Polynomial families that are computable by circuits of size poly(n) are captured by the algebraic complexity class VP (which stands for Valiant's P).

- VP is also known as AlgP/poly.

Complexity Class VP

Class VP

- A *polynomial family* $\mathcal{F} = \{f_n\}_{n\geq 1}$ is a countable set of polynomials over a field $\mathbb{F}$, i.e., f_n has coefficients in $\mathbb{F}$.

- Definition. *(Valiant '79)* A polynomial family $\mathcal{F} = \{f_n\}_{n\geq 1}$ is in class VP if there's a polynomial function $p: \mathbb{N} \to \mathbb{N}$ such that for every $n \geq 1$, f_n has number of variables as well as degree bounded by $p(n)$ and f_n is computable by a circuit of size $p(n)$.

- W.l.o.g. assume that nodes of a circuit have fan-in bounded by 2 (unless the depth is a constant).

Class VP

- A *polynomial family* $\mathcal{F} = \{f_n\}_{n\geq 1}$ is a countable set of polynomials over a field $\mathbb{F}$, i.e., f_n has coefficients in $\mathbb{F}$.

- Definition. *(Valiant '79)* A polynomial family $\mathcal{F} = \{f_n\}_{n\geq 1}$ is in class VP if there's a polynomial function $p: \mathbb{N} \to \mathbb{N}$ such that for every $n \geq 1$, f_n has number of variables as well as degree bounded by $p(n)$ and f_n is computable by a circuit of size $p(n)$.

- Valiant called such a family $\mathcal{F}$ *p-computable*.

Class VP

- A <u>*polynomial family*</u> $\mathcal{F} = \{f_n\}_{n\geq 1}$ is a countable set of polynomials over a field $\mathbb{F}$, i.e., f_n has coefficients in $\mathbb{F}$.

- Definition. *(Valiant '79)* A polynomial family $\mathcal{F} = \{f_n\}_{n\geq 1}$ is in class VP if there's a polynomial function $p: \mathbb{N} \rightarrow \mathbb{N}$ such that for every $n \geq 1$, f_n has number of variables as well as <u>degree</u> bounded by $p(n)$ and f_n is computable by a circuit of size $p(n)$.

- VP is the algebraic analogue of P/poly.
- Question. Why is there a $poly(n)$ degree bound on f_n?

Degree restriction for VP families

- The natural polynomials that we have encountered so far – Det_n, $\text{ESym}_{n,d}$, $\text{PSym}_{n,d}$ – have degrees bounded by poly(n), where n is the number of variables.

Degree restriction for VP families

- The natural polynomials that we have encountered so far – Det_n, $ESym_{n,d}$, $PSym_{n,d}$ – have degrees bounded by $poly(n)$, where n is the number of variables.

- Recall from Lecture 1 that there's a unique multilinear polynomial corresponding to every Boolean function. Thus, for Boolean circuit lower bound, it is necessary to prove arithmetic circuit lower bound computing multilinear polynomials. A multilinear polynomial in n variables has degree $\leq n$.

Degree restriction for VP families

- A circuit of size s can compute a polynomial of degree $2^{O(s)}$. We may not be able to evaluate a circuit efficiently if there's no degree restriction. For e.g., x^{2^s} can be computed a circuit of size $O(s)$. At $x=2$, x^{2^s} has exponential in s bit complexity.

Degree restriction for VP families

- A circuit of size s can compute a polynomial of degree $2^{O(s)}$. We may not be able to evaluate a circuit efficiently if there's no degree restriction. For e.g., x^{2^s} can be computed a circuit of size $O(s)$. At $x=2$, x^{2^s} has exponential in s bit complexity.

- Removal of division gates, homogenization of circuits cannot be done efficiently without a degree bound.

- We shall see later that depth reduction of circuits crucially needs a polynomial bound on degree.

Degree restriction for VP families

- For more on degree restriction for VP families:

  Ref.

  https://cstheory.stackexchange.com/questions/19261/degree-restriction-for-polynomials-in-mathsfvp

- Class VP_{nb}: Same as VP but with no bound on degree :

  Refs.

  1. *"Polynômes et coefficients", PhD Thesis, by Malod, (2003)*
  2. *"Interpolation in Valiant's Theory", by Koiran & Perifel (2007)*

Examples of families in VP

- It follows from Lecture 2 that $\mathrm{Det} = \{\mathrm{Det}_n\}_{n\geq 1}$ is in VP.
- Repeated squaring gives a circuit of size $O(n \log d)$ and depth $O(\log nd)$ for $\mathrm{PSym}_{n,d} = x_1^d + \ldots + x_n^d$. So, $\mathrm{PSym} = \{\mathrm{PSym}_{n,\mathrm{poly}(n)}\}_{n\geq 1} \in$ VP.

Examples of families in VP

- It follows from Lecture 2 that $\text{Det} = \{\text{Det}_n\}_{n\geq 1}$ is in VP.

- Repeated squaring gives a circuit of size $O(n \log d)$ and depth $O(\log nd)$ for $\text{PSym}_{n,d} = x_1^d + \ldots + x_n^d$. So, $\text{PSym} = \{\text{PSym}_{n,\text{poly}(n)}\}_{n\geq 1} \in \text{VP}$.

- Theorem. *(Baur & Strassen '83)* Any circuit computing $\text{PSym}_{n,d}$ has size $\Omega(n \log d)$.
- *Proof.* We'll see later.

Examples of families in VP

- It follows from Lecture 2 that $\text{Det} = \{\text{Det}_n\}_{n\geq 1}$ is in VP.

- Repeated squaring gives a circuit of size $O(n \log d)$ and depth $O(\log nd)$ for $\text{PSym}_{n,d} = x_1^d + \ldots + x_n^d$. So, $\text{PSym} = \{\text{PSym}_{n,\text{poly}(n)}\}_{n\geq 1} \in \text{VP}$.

- Also, observe that $\text{PSym}_{n,d}$ has a depth-2 circuit of size $O(nd)$.

Examples of families in VP

- It follows from Lecture 2 that $\mathrm{Det} = \{\mathrm{Det}_n\}_{n\geq 1}$ is in VP.

- Repeated squaring gives a circuit of size $O(n \log d)$ and depth $O(\log nd)$ for $\mathrm{PSym}_{n,d} = x_1^d + \ldots + x_n^d$. So, $\mathrm{PSym} = \{\mathrm{PSym}_{n,\mathrm{poly}(n)}\}_{n\geq 1} \in \mathrm{VP}$.

- The *sum-product polynomial* $\mathrm{SP}_{s,d} := \sum_{i\in[s]} \prod_{j\in[d]} x_{i,j}$ has sd variables, degree d, and is computable by a circuit of size $O(sd)$ and depth 2. So, $\mathrm{SP} = \{\mathrm{SP}_{s,d}\}_{s,d\geq 1}$ is in VP.

Examples of families in VP

- The <u>*iterated matrix multiplication*</u> polynomial $IMM_{w,d}$ is defined as the $(1,1)$-th entry of the product of d many $w \times w$ symbolic matrices $X_1, \ldots, X_d$, where $X_i = (x_{i,j,k})_{j,k\in[w]}$. It has $w^2(d-2) + 2w$ variables, degree d, and is computable by a circuit of size $O(w^3d)$ and depth $O(\log w \cdot \log d)$. So, $IMM = \{IMM_{w,d}\}_{w,d\geq 1}$ is in VP.

Divide and conquer

Examples of families in VP

- The <u>*iterated matrix multiplication*</u> polynomial $IMM_{w,d}$ is defined as the $(1,1)$-th entry of the product of d many $w \times w$ symbolic matrices $X_1, \ldots, X_d$, where $X_i = (x_{i,j,k})_{j,k\in[w]}$. It has $w^2(d-2) + 2w$ variables, degree d, and is computable by a circuit of size $O(w^3d)$ and depth $O(\log w \cdot \log d)$. So, $IMM = \{IMM_{w,d}\}_{w,d\geq 1}$ is in VP.

- Sometimes, $IMM_{w,d}$ is defined as $tr(X_1 \cdot \ldots \cdot X_d)$.

Examples of families in VP

- The *iterated matrix multiplication* polynomial $IMM_{w,d}$ is defined as the $(1,1)$-th entry of the product of d many $w \times w$ symbolic matrices $X_1, \ldots, X_d$, where $X_i = (x_{i,j,k})_{j,k \in [w]}$. It has $w^2(d-2) + 2w$ variables, degree d, and is computable by a circuit of size $O(w^3 d)$ and depth $O(\log w \cdot \log d)$. So, $IMM = \{IMM_{w,d}\}_{w,d \geq 1}$ is in VP.

- Sometimes, $IMM_{w,d}$ is defined as $tr(X_1 \cdot \ldots \cdot X_d)$.

- Is $ESym = \{ESym_{n,d}\}_{n,d \geq 1}$ in VP?

Examples of families in VP

- The *iterated matrix multiplication* polynomial $IMM_{w,d}$ is defined as the $(1,1)$-th entry of the product of d many $w \times w$ symbolic matrices $X_1, \ldots, X_d$, where $X_i = (x_{i,j,k})_{j,k\in[w]}$. It has $w^2(d-2) + 2w$ variables, degree d, and is computable by a circuit of size $O(w^3d)$ and depth $O(\log w \cdot \log d)$. So, $IMM = \{IMM_{w,d}\}_{w,d\geq 1}$ is in VP.

- Sometimes, $IMM_{w,d}$ is defined as $tr(X_1 \cdot \ldots \cdot X_d)$.

- Is $ESym = \{ESym_{n,d}\}_{n,d\geq 1}$ in VP? Yes. Let's see why…

Circuits computing ESym

Circuits for ESym over char 0 fields

- Theorem. $\mathrm{ESym}_{n,d}$ can be computed by a circuit of size $O(nd)$ provided $\mathrm{char}(\mathbb{F}) = 0$ or $> d$.
- *Proof.* From Newton-Gerard identities (Lecture 2), and Cramer's rule,

$$\mathrm{ESym}_{n,d} = 1/d! \cdot \det \underbrace{\begin{pmatrix} p_1 & 1 & 0 & \cdots & 0 \\ p_2 & p_1 & 2 & \cdots & 0 \\ p_3 & p_2 & p_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_d & p_{d-1} & \cdots & p_2 & p_1 \end{pmatrix}}_{M}$$

Circuits for ESym over char 0 fields

- Theorem. $ESym_{n,d}$ can be computed by a circuit of size $O(nd)$ provided $char(\mathbb{F}) = 0$ or $> d$.
- *Proof.* From Newton-Gerard identities (Lecture 2),
- $ESym_{n,d} = 1/d! \cdot det(M)$.
- Obs. $p_1,\ldots,p_d$ can be computed by a circuit of size $O(nd)$ and depth $O(\log nd)$ (*why?*).
- Hence, by Csanky's theorem, $det(M)$ can be computed by a circuit of size $O(nd + d^4)$ and depth $O(\log nd + (\log d)^2)$.

Circuits for ESym over char 0 fields

- Theorem. $ESym_{n,d}$ can be computed by a circuit of size $O(nd)$ provided $char(\mathbb{F}) = 0$ or $> d$.
- *Proof.* From Newton-Gerard identities (Lecture 2),
- $ESym_{n,d} = 1/d! \cdot det(M)$.
- Obs. $p_1,\ldots,p_d$ can be computed by a circuit of size $O(nd)$ and depth $O(\log nd)$ (*why?*).

- *Homework.* Once $p_1,\ldots,p_d$ are computed, $det(M)$ can be computed by a circuit of size $O(d^2)$. (Use the special structure of M.)

Circuits for ESym over char 0 fields

- Theorem. $ESym_{n,d}$ can be computed by a circuit of size $O(nd)$ provided $char(\mathbb{F}) = 0$ or $> d$.
- *Proof.* From Newton-Gerard identities (Lecture 2),
- $ESym_{n,d} = \quad 1/d! \cdot \det(M)$.
- Obs. $p_1,\ldots,p_d$ can be computed by a circuit of size $O(nd)$ and depth $O(\log nd)$ (*why?*).
- Therefore, $ESym_{n,d}$ can be computed by a circuit of size $O(nd)$ provided $char(\mathbb{F}) = 0$ or $> d$.

∎

Circuits for ESym over char 0 fields

- Theorem. $ESym_{n,d}$ can be computed by a circuit of size $O(nd)$ provided $char(\mathbb{F}) = 0$ or $> d$.

- The merit of this proof is that it yields a circuit of subquadratic size and low depth if d is small, e.g., if $d = n^{1/3}$, the circuit has size $O(n^{4/3})$ and depth $O((\log n)^2)$.

- Question. What about circuits over fields of low char.?

Circuits for ESym over any field

- Theorem. $\mathrm{ESym}_{n,d}$ can be computed by a circuit of size $O(nd)$ over *any* field.
- *Proof.* Denote $\mathrm{ESym}_{n,k}$ as $e_{n,k}$. Observe that

$$
\begin{aligned}
e_{n,k} &= e_{n-1,k} + x_n \cdot e_{n-1,k-1} \\
&= e_{n-2,k} + x_{n-1} \cdot e_{n-2,k-1} + x_n \cdot e_{n-1,k-1} \\
&\vdots \\
&= x_k \cdot e_{k-1,k-1} + x_{k+1} \cdot e_{k,k-1} + \ldots + x_{n-1} \cdot e_{n-2,k-1} + x_n \cdot e_{n-1,k-1}
\end{aligned}
$$

Circuits for ESym over any field

- Theorem. $\mathrm{ESym}_{n,d}$ can be computed by a circuit of size $O(nd)$ over *any* field.
- *Proof.* Denote $\mathrm{ESym}_{n,k}$ as $e_{n,k}$. Observe that

$$
\begin{aligned}
e_{n,k} &= e_{n-1,k} + x_n \cdot e_{n-1,k-1} \\
&= e_{n-2,k} + x_{n-1} \cdot e_{n-2,k-1} + x_n \cdot e_{n-1,k-1} \\
&\vdots \\
&= x_k \cdot e_{k-1,k-1} + x_{k+1} \cdot e_{k,k-1} + \ldots + x_{n-1} \cdot e_{n-2,k-1} + x_n \cdot e_{n-1,k-1}
\end{aligned}
$$

- This suggests the following dynamic programming approach: For $k \in [2,d]$, compute $e_{k-1,k-1}, \ldots, e_{n-1,k-1}$; then compute $x_k \cdot e_{k-1,k-1}, \ldots, x_n \cdot e_{n-1,k-1}$. From these compute $e_{k,k}, \ldots, e_{n,k}$ using $O(n)$ multiplications and additions.

Circuits for ESym over any field

- Theorem. $ESym_{n,d}$ can be computed by a circuit of size $O(nd)$ over *any* field.
- *Proof.* Denote $ESym_{n,k}$ as $e_{n,k}$. Observe that

$$
\begin{aligned}
e_{n,k} &= e_{n-1,k} + x_n \cdot e_{n-1,k-1} \\
&= e_{n-2,k} + x_{n-1} \cdot e_{n-2,k-1} + x_n \cdot e_{n-1,k-1} \\
&\vdots \\
&= x_k \cdot e_{k-1,k-1} + x_{k+1} \cdot e_{k,k-1} + \ldots + x_{n-1} \cdot e_{n-2,k-1} + x_n \cdot e_{n-1,k-1}
\end{aligned}
$$

- Thus, we use a total of $O(nd)$ multiplications and additions. However, the depth of the circuit is $O(d)$. The construction works over *any* field. ■

Circuits for ESym over any field

- Theorem. $ESym_{n,d}$ can be computed by a circuit of size $O(nd)$ over *any* field.
- Two important features of the circuit are:

1. It is *monotone*, i.e., there's no negation, and so, no cancellation of monomials generated in the circuit.
2. It is *skew*, i.e., every × gate has at most one child that is not a leaf node.

Circuits for ESym over any field

- Theorem. $\mathrm{ESym}_{n,d}$ can be computed by a *monotone*, *skew* circuit of size $O(nd)$ & depth $O(d)$ over *any* field.
- Two important features of the circuit are:

1. It is *monotone*, i.e., there's no negation, and so, no cancellation of monomials generated in the circuit.
2. It is *skew*, i.e., every × gate has at most one child that is not a leaf node.

- We'll see later that skew circuits form an important subclass of circuits, namely **Algebraic Branching Programs**. We'll use ABPs to define the class VBP.

Circuits for ESym over any field

- Theorem. $ESym_{n,d}$ can be computed by a _monotone_, _skew_ circuit of size $O(nd)$ & depth $O(d)$ over _any_ field.
- Two important features of the circuit are:

1. It is _monotone_, i.e., there's no negation, and so, no cancellation of monomials generated in the circuit.
2. It is _skew_, i.e., every $\times$ gate has at most one child that is not a leaf node.

- Question. Can $ESym_{n,d}$ be computed by a constant depth circuit (like $PSym_{n,d}$)?
- A small depth-2 circuit is not possible as ESym has too many monomials. How about a depth-3 circuit?

Depth-3 circuits computing ESym

- Theorem. *(Ben-Or)* $\mathrm{ESym}_{n,d}$ can be computed by a depth-3 circuit of size $O(n^2)$ provided $|\mathbb{F}| > n$.
- *Proof.* Observe that

$$f(\mathbf{x}, y) := (1+x_1 y) \cdot \ldots \cdot (1+x_n y)$$
$$= 1 + \mathrm{ESym}_{n,1}(\mathbf{x})y + \ldots + \mathrm{ESym}_{n,n}(\mathbf{x})y^n$$

- The idea is to use *polynomial interpolation*.

- Let $\alpha_1, \ldots, \alpha_{n+1}$ be distinct elements of $\mathbb{F}$, and V be the Vandermonde matrix $(\alpha_i^j)_{i\in[n+1],\, j\in[0,n]}$.

Depth-3 circuits computing ESym

- Theorem. *(Ben-Or)* $\mathrm{ESym}_{n,d}$ can be computed by a <u>depth-3 circuit</u> of size $O(n^2)$ provided $|\mathbb{F}| > n$.
- *Proof.* Then,

$$\begin{pmatrix} 1 \\ \mathrm{ESym}_{n,1} \\ \mathrm{ESym}_{n,2} \\ \vdots \\ \mathrm{ESym}_{n,n} \end{pmatrix} = V^{-1} \cdot \begin{pmatrix} f(\mathbf{x}, \alpha_1) \\ f(\mathbf{x}, \alpha_2) \\ f(\mathbf{x}, \alpha_3) \\ \vdots \\ f(\mathbf{x}, \alpha_{n+1}) \end{pmatrix}$$

Depth-3 circuits computing ESym

- Theorem. *(Ben-Or)* $\mathrm{ESym}_{n,d}$ can be computed by a depth-3 circuit of size $O(n^2)$ provided $|\mathbb{F}| > n$.
- *Proof.* Thus,

$$\mathrm{ESym}_{n,d} = \beta_{d,1} f(\mathbf{x}, \alpha_1) + \ldots + \beta_{d,n+1} f(\mathbf{x}, \alpha_{n+1}).$$

- $\beta_{d,1}, \ldots, \beta_{d,n+1}$ are $\mathbb{F}$-constants dependent only on $\alpha_1, \ldots, \alpha_{n+1}$.
- The above expression gives a depth-3 circuit of size $O(n^2)$ and top fan-in $n+1$ for $\mathrm{ESym}_{n,d}$ for every d. ∎

Depth-3 circuits computing ESym

- Theorem. *(Ben-Or)* $\mathrm{ESym}_{n,d}$ can be computed by a depth-3 circuit of size $O(n^2)$ provided $|\mathbb{F}| > n$.

- Question. Does $\mathrm{ESym}_{n,d}$ have a depth-3 circuit of size poly(n) over *fixed* finite fields for every d?

Depth-3 circuits computing ESym

- Theorem. *(Ben-Or)* $ESym_{n,d}$ can be computed by a depth-3 circuit of size $O(n^2)$ provided $|\mathbb{F}| > n$.

- Question. Does $ESym_{n,d}$ have a depth-3 circuit of size poly(n) over *fixed* finite fields for every d? No!
- Can be proved using methods by *Grigoriev & Karpinski (1998)* and *Grigoriev & Razborov (1998)*.

- Ref. See Theorem 10.2 in the survey *https://github.com/dasarpmar/lowerbounds-survey/releases/download/v9.0.3/fancymain.pdf*

Depth-3 circuits computing ESym

- Theorem. *(Ben-Or)* $ESym_{n,d}$ can be computed by a depth-3 circuit of size $O(n^2)$ provided $|\mathbb{F}| > n$.
- Question. Does $ESym_{n,d}$ have a depth-3 circuit of size poly(n) over *fixed* finite fields for every d? No!
- Question. Does $ESym_{n,d}$ have a *constant depth* circuit of size poly(n) over *fixed* finite fields?
- **We do not know.**

Almost linear size circuit for ESym

- Theorem. *(Ben-Or)* $ESym_{n,d}$ can be computed by a circuit of size $O(n(\log d)^2)$ over complex numbers.

- The proof uses Fast Fourier Transform (FFT) for polynomial multiplication.

- Ref. See the first answer to the post *https://cstheory.stackexchange.com/questions/33503/monotone-arithmetic-circuit-complexity-of-elementary-symmetric-polynomials*

Almost linear size circuit for ESym

- Theorem. *(Ben-Or)* $\mathrm{ESym}_{n,d}$ can be computed by a circuit of size $O(n(\log d)^2)$ over complex numbers.

- Theorem. *(Baur & Strassen '83)* Any circuit computing $\mathrm{ESym}_{n,n/2}$ has size $\Omega(n \log n)$.

ABPs and class VBP

Algebraic Branching Programs

- Definition. An *algebraic branching program* (ABP) B is a directed acyclic graph with a source node s and a sink node t. The edges are labelled by *affine forms* in $x_1, \ldots, x_n$ variables. The *weight* of a path is the product of the labels of the edges in the path. The polynomial computed by a node v is the *sum of the weights* of all paths from s to v. The polynomial computed by B is the one computed by the sink node t.

  The *size* of B is the number of edges in it.

  The *length* of B is the length of the longest path from s to t.

Algebraic Branching Programs

- Definition. An *algebraic branching program* (ABP) B is a directed acyclic graph with a source node s and a sink node t. The edges are labelled by *affine forms* in $x_1, \ldots, x_n$ variables. The *weight* of a path is the product of the labels of the edges in the path. The polynomial computed by a node v is the *sum of the weights* of all paths from s to v. The polynomial computed by B is the one computed by the sink node t.

  The *size* of B is the number of edges in it.

  The *length* of B is the length of the longest path from s to t. (Obs. The polynomial computed by B has degree at most the length of B.)

Algebraic Branching Programs

- Definition. An *algebraic branching program* (ABP) B is a directed acyclic graph with a source node s and a sink node t. The edges are labelled by *affine forms* in $x_1, \ldots, x_n$ variables. The *weight* of a path is the product of the labels of the edges in the path. The polynomial computed by a node v is the *sum of the weights* of all paths from s to v. The polynomial computed by B is the one computed by the sink node t.

  An ABP B is *layered* if the nodes can be partitioned into layers $V_0, \ldots, V_d$, with $V_0 = \{s\}$ and $V_d = \{t\}$, such that every edge is incident between a node in V_i and a node in V_{i+1} for some $i \in [0,d-1]$.

Algebraic Branching Programs

- Definition. An *algebraic branching program* (ABP) B is a directed acyclic graph with a source node s and a sink node t. The edges are labelled by *affine forms* in $x_1, \ldots, x_n$ variables. The *weight* of a path is the product of the labels of the edges in the path. The polynomial computed by a node v is the *sum of the weights* of all paths from s to v. The polynomial computed by B is the one computed by the sink node t.

  The *width* of a layered ABP B with layers $V_0, \ldots, V_d$ is $\max_i\{|V_i|\}$.

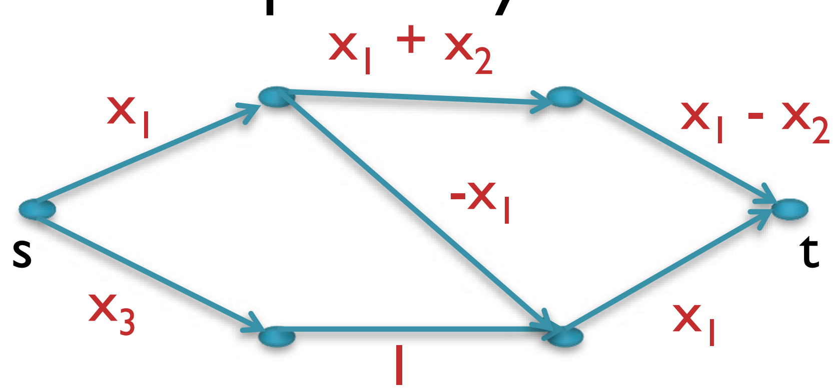
Algebraic Branching Programs

- Definition. An *algebraic branching program* (ABP) B is a directed acyclic graph with a source node s and a sink node t. The edges are labelled by *affine forms* in $x_1, \ldots, x_n$ variables. The *weight* of a path is the product of the labels of the edges in the path. The polynomial computed by a node v is the *sum of the weights* of all paths from s to v. The polynomial computed by B is the one computed by the sink node t.

  Obs. The polynomial computed by a layered ABP B with layers $V_0, \ldots, V_d$ has degree at most d.

Algebraic Branching Programs

- Definition. An *algebraic branching program* (ABP) B is a directed acyclic graph with a source node s and a sink node t. The edges are labelled by *affine forms* in $x_1, \ldots, x_n$ variables. The *weight* of a path is the product of the labels of the edges in the path. The polynomial computed by a node v is the *sum of the weights* of all paths from s to v. The polynomial computed by B is the one computed by the sink node t.



The layered ABP in the figure computes $x_1x_3 - x_1x_2^2$. Its size is 7 and length is 3.

Algebraic Branching Programs

- Definition. An *algebraic branching program* (ABP) B is a directed acyclic graph with a source node s and a sink node t. The edges are labelled by *affine forms* in $x_1, \ldots, x_n$ variables. The *weight* of a path is the product of the labels of the edges in the path. The polynomial computed by a node v is the *sum of the weights* of all paths from s to v. The polynomial computed by B is the one computed by the sink node t.

- Obs. An ABP of size s and length d can be converted to a layered ABP of size at most sd (simply by splitting an edge into at most d edges).

Algebraic Branching Programs

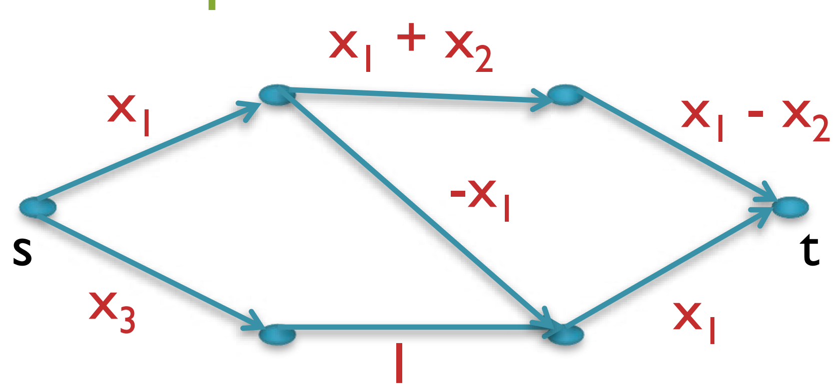
- Definition. An *algebraic branching program* (ABP) B is a directed acyclic graph with a source node s and a sink node t. The edges are labelled by *affine forms* in $x_1, \ldots, x_n$ variables. The *weight* of a path is the product of the labels of the edges in the path. The polynomial computed by a node v is the *sum of the weights* of all paths from s to v. The polynomial computed by B is the one computed by the sink node t.

- Typically, when we talk about an ABP, we mean a layered ABP.

Layered ABP & Matrix Multiplication

- Obs. A layered ABP, with layers $V_0, \ldots, V_d$, can be **equivalently** viewed as a *sequence of matrix multiplications* $M_1 \cdot M_2 \cdot \ldots \cdot M_d$, where M_i is a $|V_{i-1}| \times |V_i|$ matrix whose entries are affine forms.

- Example.



$$\begin{pmatrix} x_1 & x_3 \end{pmatrix} \cdot \begin{pmatrix} x_1 + x_2 & -x_1 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 - x_2 \\ x_1 \end{pmatrix}$$

$$= x_1x_3 - x_1x_2^2$$

Layered ABP & Matrix Multiplication

- Obs. A layered ABP, with layers $V_0, \ldots, V_d$, can be **equivalently** viewed as a *sequence of matrix multiplications* $M_1 \cdot M_2 \cdot \ldots \cdot M_d$, where M_i is a $|V_{i-1}| \times |V_i|$ matrix whose entries are affine forms.

- Corollary. An n-variate polynomial computable by a layered ABP of width w and length d can be computed by a circuit of size $O(w^2 nd + w^3 d)$ & depth $O(\log w \cdot \log d)$.

Homogenization of ABP

- Definition. An ABP is *homogeneous* if every node of the ABP computes a homogeneous polynomial.

- Obs. Let p be a degree-d homogeneous polynomial that is computable by a size-s ABP. Then, p is also computable by a homogeneous ABP of size $O(ds)$.
- *Proof sketch.* For every node v computing f, create nodes $v_0, \ldots, v_d$ that compute $f^{[0]}, f^{[1]}, \ldots, f^{[d]}$.

- Recall from Lecture 1 that homogenization of circuits can also be done efficiently.

Class VBP

- Definition. A polynomial family $\mathcal{F} = \{f_n\}_{n\geq 1}$ is in class VBP if there's a polynomial function $p: \mathbb{N} \to \mathbb{N}$ s.t. for every $n \geq 1$, f_n has number of variables bounded by $p(n)$ and f_n is computable by an ABP of size $p(n)$.

Class VBP

- Definition. A polynomial family $\mathcal{F} = \{f_n\}_{n\geq 1}$ is in class VBP if there's a polynomial function $p: \mathbb{N} \to \mathbb{N}$ s.t. for every $n \geq 1$, f_n has number of variables bounded by $p(n)$ and f_n is computable by an ABP of size $p(n)$.

- Why is there **no** degree restriction in the above definition? (unlike the definition of class VP)
- That's because the degree of the polynomial computed by an ABP B is bound by the length of B which in turn is bounded by the size of B.

Class VBP

- Definition. A polynomial family $\mathcal{F} = \{f_n\}_{n\geq 1}$ is in class VBP if there's a polynomial function $p: \mathbb{N} \to \mathbb{N}$ s.t. for every $n \geq 1$, f_n has number of variables bounded by $p(n)$ and f_n is computable by an ABP of size $p(n)$.

- It follows from the last corollary that VBP $\subseteq$ VP.

- Question. Is VBP strictly contained in VP?
- **We do not know.**

Examples of families in VBP

- Obs. The families IMM, PSym and SP are in VBP.
- *Proof.* Easy exercise.

Examples of families in VBP

- Obs. The families IMM, PSym and SP are in VBP.
- *Proof.* Easy exercise.

- Theorem. Det is in VBP.
- *Proof sketch.* Csanky's algorithm gives an ABP of size $O(n^6)$ for Det_n over fields of characteristic 0 or $> n$. Use Equation 1 in Lecture 2: Compute each p_i using an ABP of size $O(n^4)$. Compute the entries of P using an ABP of size $O(n^5)$. Finally, compute $(I_n + P)^{-1}$ using an ABP of size $O(n^6)$. (*Homework:* Fill in the details.) ∎

Examples of families in VBP

- Obs. The families IMM, PSym and SP are in VBP.
- *Proof.* Easy exercise.

- Theorem. Det is in VBP.
- Berkowitz's algorithm gives a *poly(n)* ($O(n^{18})$?) size ABP for Det_n over any field.

- *Mahajan & Vinay (1997)* gave an $O(n^6)$ size ABP computing Det_n over any field by proving a combinatorial characterization of the determinant.

Examples of families in VBP

- Obs. The families IMM, PSym and SP are in VBP.
- *Proof.* Easy exercise.

- Theorem. Det is in VBP.

- Question. Is ESym in VBP?
- Yes, it is. The depth-3 circuit for $\text{ESym}_{n,d}$ gives an ABP of size $O(n^2)$ and depth n, provided $|\mathbb{F}| > n$.
- The skew circuit construction for $\text{ESym}_{n,d}$ gives an ABP of size $O(nd)$ over *any* field.

Skew circuits and ABPs

- Obs. Skew circuits are *essentially* ABPs.
- *Proof sketch.* If a polynomial is computed by an ABP of size s then it can also be computed by a skew circuit of size O(ns). Conversely, a skew circuit of size s computing a polynomial gives an ABP of size O(s) computing the same polynomial. (*Homework:* Fill in the details.)

∎

Skew circuits and ABPs

- Obs. Skew circuits are *essentially* ABPs.
- *Proof sketch.* If a polynomial is computed by an ABP of size s then it can also be computed by a skew circuit of size $O(ns)$. Conversely, a skew circuit of size s computing a polynomial gives an ABP of size $O(s)$ computing the same polynomial. (*Homework:* Fill in the details.)

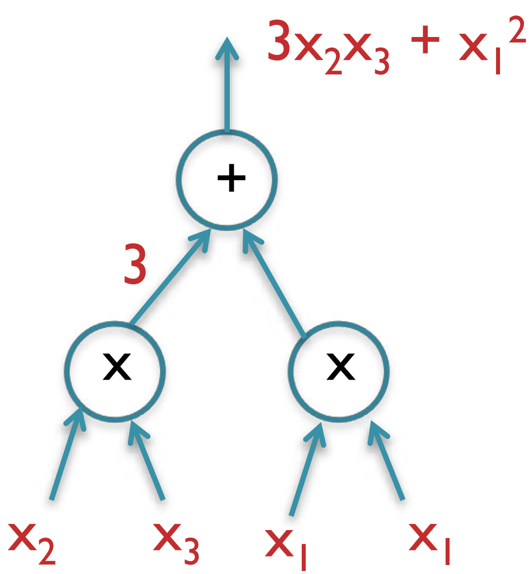
  ■

- Thus VBP can be equivalently defined as the class of families of polynomials computable by polynomial size skew circuits.

Formulas and class VF

Arithmetic formulas

- Definition. An *arithmetic formula* is a circuit whose underlying graph is a tree. In other words, the out-degree of every node is at most one in a formula.



Size = 7
Depth = 2

Arithmetic formulas

- Definition. An *arithmetic formula* is a circuit whose underlying graph is a tree. In other words, the out-degree of every node is at most one in a formula.

- Obs. An n-variate polynomial computable by a formula of size s can be computed by an ABP of size s.
- *Proof sketch.* Induct on the size of the formula: If a node of the formula computes $f_1 + f_2$, attach the ABPs computing f_1 and f_2 in parallel. If a node computes $f_1 \cdot f_2$, attach the corresponding ABPs in series. ∎

Class VF

- Definition. A polynomial family $\mathcal{F} = \{f_n\}_{n\geq 1}$ is in class VF if there's a polynomial function $p: \mathbb{N} \rightarrow \mathbb{N}$ s.t. for every $n \geq 1$, f_n has number of variables bounded by $p(n)$ and f_n is computable by a formula of size $p(n)$.

- Why is there **no** degree restriction in the above definition? (unlike the definition of class VP)
- Obs. A formula of size s computes a polynomial of degree at most s.
- *Proof sketch.* Can be proved by inducting on size. ◼

Class VF

- Definition. A polynomial family $\mathcal{F} = \{f_n\}_{n \geq 1}$ is in class VF if there's a polynomial function $p: \mathbb{N} \to \mathbb{N}$ s.t. for every $n \geq 1$, f_n has number of variables bounded by $p(n)$ and f_n is computable by a formula of size $p(n)$.

- It follows from a previous Obs that VF $\subseteq$ VBP.

- Question. Is VF strictly contained in VBP?
- **We do not know.**

Examples of families in VF

- Obs. The families PSym and SP are in VF.
- *Proof.* Easy exercise.

- Obs. The family ESym is in VF over infinite fields.
- *Proof.* Ben-Or's construction of a depth-3 circuit.

Examples of families in VF

- Obs. The families PSym and SP are in VF.
- *Proof.* Easy exercise.

- Obs. The family ESym is in VF over infinite fields.
- *Proof.* Ben-Or's construction of a depth-3 circuit.
- Question. Is ESym in VF over any field?
- **We do not know.**

Examples of families in VF

- Obs. The families PSym and SP are in VF.
- *Proof.* Easy exercise.

- Obs. The family ESym is in VF over infinite fields.
- *Proof.* Ben-Or's construction of a depth-3 circuit.
- Question. Is ESym in VF over any field?
- **We do not know.**

- Question. Are the familes Det and IMM in VF?
- **We do not know.** We'll see that if yes then VBP = VF.