Algebraic Computation

Tzovas Charilaos

MPLA

January 26, 2015

Motivation

Our motivation is to investigate the complexity of algebraic problems, e.g:

- The common root of a system of polynomials.
- The permanent or determinant of an $n \times n$ matrix.

Problems that arise in computational geometry, numeric analysis etc. that involve operations $(+, \times, \div)$ in some field $\mathbb{F}$.

Preliminaries

- We are refering to some field $\mathbb{F}$, such as the real or the complex numbers.
- The input to our problems will be $(x_1, x_2 \ldots, x_n) \in \mathbb{F}^n$ of size n.
- But can we implement real numbers in computers?

Preliminaries

- We are refering to some field $\mathbb{F}$, such as the real or the complex numbers.
- The input to our problems will be $(x_1, x_2 \ldots, x_n) \in \mathbb{F}^n$ of size n.
- But can we implement real numbers in computers?
  - We are looking mostly for lower bounds, so we can consider stronger, unrealistic models.
- We will now define appropriate models to measure complexity of these problems .

Straight-Line Programs

Definition

An *algebraic straight line program* of length T with input variables $x_1, x_2, \ldots, x_n \in \mathbb{F}$ and built-in constants $c_1, c_2, \ldots, c_k \in \mathbb{F}$ is a sequence of T statements of the form $y_i = z_{i1}\, OP\, z_{i2}$ for $i = 1, 2, \ldots, T$ where OP is one of the field operations $\times$ or $+$ and each z_{i1}, z_{i2} is either an input variable, or a built-in constant, or some y_j for $j < i$.
For every setting of values to the input variables, the straight line computation consists of executing these simple statements in order and find the values $y_1, y_2, \ldots, y_T$. The *output* of the computation is the value y_T.

Straight-Line Programs

- They look like programs without any branching or loops, just simple assignments.
- A straight line program for the equation $f(x_1, x_2, x_3) = (x_1 + x_2)^2(x_3 + 4) + 2x_2$ would be :

$$
\begin{aligned}
y_1 &= x_1 + x_2 \\
y_2 &= y_1 \times y_1 \\
y_3 &= x_3 + 4 \\
y_4 &= y_3 \times y_2 \\
y_5 &= x_2 + x_2 \\
y_6 &= y_4 + y_5
\end{aligned}
$$

Straight-Line Programs

- The asymptotic complexity is the *length* of the shortest family of algebraic straight-line programs that compute a family of functions $\{f_n\}$ where f_n is a function of n variables.
- Straight-line programs over $\{0, 1\}$ are equivalent to Boolean circuits.
- Functions computable by polynomial length straight-line programs:
  - Fast Fourier Transformation , Matrix Multiplication, Determinant.

Straight-Line Programs

Every straight line program computes a polynomial of degree related to its length.

Proposition

The output of a straight line program of length T with variables $x_1, x_2, \ldots, x_n$ is a polynomial $p(x_1, x_2, \ldots, x_n)$ of degree at most 2^T.

We can reach degree 2^T only if we quadrate a single variable T times.

- What if we allow division $(\div)$?

Straight-Line Programs

Every straight line program computes a polynomial of degree related to its length.

Proposition

The output of a straight line program of length T with variables $x_1, x_2, \ldots, x_n$ is a polynomial $p(x_1, x_2, \ldots, x_n)$ of degree at most 2^T.

We can reach degree 2^Tonly if we quadrate a single variable T times.

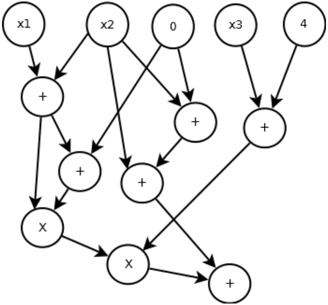
- What if we allow division $(\div)$?
- Strassen proved that if a program uses division we can transform it to an equivalent program that does not uses this operator and has similar size.

Blum-Shub-Smale Model

- Generalization of a Turing Machine where each cell can hold an element from a field $\mathbb{F}$.
- *Shift* and *Branch* states. Branch tests check only for equallity, not inequality.
    - If we allow inequallity tests we could decide every language in $\mathrm{P}_{/\mathrm{poly}}$.
- *Computation* states associated with a hard-wired function f that computes and stores $f(a)$ for some value a.

Algebraic Circuits

An *algebraic circuit* is defined by analogy with a Boolean circuit. The inputs nodes accept values from $\mathbb{F}$ and the internal nodes, the *gates*, are labeled with one operation $\{+, \times\}$. The circuit has one output gate and each gate has fan-in 2.



$$f(x_1, x_2, x_3) = (x_1 + x_2)^2(x_3 + 4) + 2x_2$$

Algebraic Circuits

Definitions

1. The *size* of the circuit is the number of operational gates.
2. The *depth* of the circuit is length of the longest path.

Straight-line programs and algebraic circuits are equivalent:

Proposition

Let $f : \mathbb{F}^n \to \mathbb{F}$. If f has an algebraic straight-line program of size S then it has an algebraic circuit of size $3S$. If f is computable by an algebraic circuit of size S then it is computable by an algebraic straight-line program of length S.

The analogues of P and NP

Definition (Valiant 1979)

A family of polynomials $\{p_n\}$ over $\mathbb{F}$ has *polynomially bounded degree* if $\exists c$ constant, $\forall n \in \mathbb{N}$ the degree of p_n is at most cn^c.

- The class $\mathrm{AlgP}_{/\mathrm{poly}}$ (or VP) contains all polynomially bounded degree families $\{p_n\}$ of polynomials that are computable by algebraic circuits of polynomial size.
- The class $\mathrm{AlgNP}_{/\mathrm{poly}}$ (or VNP) is the class of polynomially bounded degree families $\{p_n\}$ that are definable as

$$p_n(x_1, x_2, \ldots, x_n) = \sum_{e \in \{0,1\}^{m-n}} g_m(x_1, x_2, \ldots, x_n, e_{n+1}, \ldots, e_m)$$

where $g_m \in \mathrm{AlgP}_{/\mathrm{poly}}$ and m is polynomial in n.

On VNP

On the definition of VNP.

- The idea is to define VNP similar to NP.
  - $A \in \mathrm{NP}$ if exists $B \in \mathrm{P}$ such that $x \in A \iff \exists e$ such that $(x, e) \in B$.
- $+$ is the algebraic analog of Boolean OR.
  - The definition of NP involves $\exists_{e \in \{0,1\}^{m-n}}$ that means $\bigvee_{e \in \{0,1\}^{m-n}}$ and the algebraic analog is $\sum_{e \in \{0,1\}^{m-n}}$.

Reductions

Definition (*Projection reduction*)

A function $f(x_1, x_2, \ldots, x_n)$ is a *projection* of a function $g(y_1, y_2, \ldots, y_m)$ if there is a mapping σ from $\{y_1, y_2, \ldots, y_m\}$ to $\{0, 1, x_1, x_2, \ldots, x_n\}$ such that
$f(x_1, x_2, \ldots, x_n) = g(\sigma(y_1), \sigma(y_2), \ldots, \sigma(y_m))$.
We say that f is *projection-reducable* to g if f is a projection of g.

Example

$f(x_1, x_2) = x_1 + x_2$ is projection reducible to $g(y_1, y_2, y_3) = y_1^2 y_3 + y_2$ since $f(x_1, x_2) = g(1, x_1, x_2)$.

If we have g we just hardwire its inputs to $0, 1$ or x_i and we can compute f.

Determinant and Permanent

Definition

Determinant of an $n \times n$ matrix $X = (X_{i,j})$ is defined:

$$det(X) = \sum_{\sigma \in S_n} sgn(\sigma) \prod_{i=1}^{n} X_{i,\sigma(i)}$$

Permanent is defined:

$$perm(X) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} X_{i,\sigma(i)}$$

where S_n is the set of all $n!$ permutations on $\{1, 2, \ldots, n\}$ and $sgn(\sigma) = \{-1, 1\}$ is the signature of σ.

Completeness

Theorem (Completeness of determinant and permanent (Valiant 1979))

- *Every polynomial family $\{p_n\}$ over any field $\mathbb{F}$ that is computable by a circuit if size u is projection reducible to the determinant function on $u+2$ variables.*
- *Every polynomial family $\{p_n\}$ in VNP is projection reducible to the permanent function with polynomially more variables.*

- In other words, *deteminant* $\in \mathrm{VP}$ and *permanent* $\in \mathrm{VNP} -$ *complete*.
- Obviously $\mathrm{VP} \subseteq \mathrm{VNP}$ and is an open question if $\mathrm{VP} = \mathrm{VNP}$ or not. It is believed that are not equal.

VP=VNP implications

- If one show that permanent has polynomial size algebraic circuits will prove that $\mathrm{VP} = \mathrm{VNP}$.

Conjecture (Valiant)

The $n \times n$ permanent cannot be obtained as a projection of the $m \times m$ determinant where $m = 2^{O(\log^2 n)}$.

- The Cook reduction is a projection reduction and we know that $\mathrm{VP} \stackrel{?}{=} \mathrm{VNP}$ must be answered before $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$.
- If equality is proven would imply that the polynomial hierarchy collapses to the second level.
  - Under GRY: $\mathrm{VP} = \mathrm{VNP} \implies \mathrm{P}_{/\mathrm{poly}} = \mathrm{NP}_{/\mathrm{poly}}$.
  - Also, if $\mathrm{VP} = \mathrm{VNP}$ over $\mathbb{Q}$ then $\mathrm{P}^{\#\mathrm{P}} \subset \mathrm{P}_{/\mathrm{poly}}$.

Computation Trees

- We will introduce a more powerful model, the *algebraic computation trees*.
- These are augmented straight-line programs with division and branching when a variable y_i is greater than zero or not.
- You can use the model to solve decision problems on real inputs $f : \mathbb{R}^n \to \{0, 1\}$.

Computation Trees

Definition (Algebraic Computation Tree over $\mathbb{R}$)

It is a binary tree on input vector $(x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ for a function f where each of the nodes is one of the following:

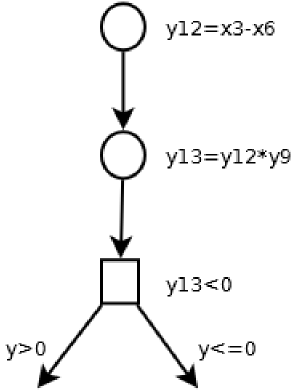
- Leaf labeled "Accept" or "Reject".
- Computational node $y_v = y_u OP\, y_w$, where $y_i = x_j$ *or* $y_{k<i}$ and $OP = \{+, -, \times, \div, \sqrt{}\}$.
- Branch node with out degree 2 upon a condition of the type $y_v = 0$, $y_v \geq 0$ *or* $y_v \leq 0$.

Computation Trees

Definition (Algebraic Computation Tree over $\mathbb{R}$ (contd))

The computation follows a single path from the root to a leaf evaluating functions at internal nodes. It reaches an "Accept" iff $f(x_1, x_2, \ldots, x_n) = 1$. The complexity is measured using the following costs.

- $+, -$ are free.
- $\times, \div, \surd$ and branch nodes are charged one unit cost.
- The *depth* of the tree is the maximum cost of any path in it.



The power of the model

Algebraic computation trees are much stronger than real life programs. A depth d algebraic computation tree would yield a classical algorithm of size 2^d since a tree can have 2^d nodes. We are going to use this model to investigate lower bounds.

Theorem (Meyer auf der Heide 1988)

The real-number version of SUBSET SUM can be solved using an algebraic computation tree of depth $O(n^5)$.

Definition (Algebraic computation tree complexity)

Let $f : \mathbb{R}^n \to \{0, 1\}$, the *algebraic computation tree complexity* of f is:

$$AC(f) = \min_{\forall tree\ T\ computes\ f} \{depth\ of\ T\}$$

Topological method

- We will try to prove lower bounds for this model.
- Using the topology of the sets $f^{-1}(0)$ and $f^{-1}(1)$. Specifically, the number of connected components.
- We say $W \subseteq \mathbb{R}^n$ is *connected* if $\forall x, y \in W$ there is a path from x to y inside W. A *connected component* of W is a maximal connected subset of W. By $\#(W)$ we denote the number of connected components of W.

Theorem (M.Ben-Or 1983)

For every $f : \mathbb{R}^n \to \{0, 1\}$,

$$AC(f) = \Omega\left(log\left(max\left\{\#(f^{-1}(1), \#(\mathbb{R}^n \setminus f^{-1}(1)\right\}\right) - n\right)$$

Lower Bounds

- Proving lower bounds for the problem `Element Distinctness`: Given n numbers $x_1, x_2, \ldots, x_n$ determine whether they are all distinct. Equivalently, determine if $\prod_{i \neq j}(x_i - x_j) \neq 0$.
- The next and Ben-Or's theorems prove the lower bound: $\Omega(nlogn)$.

Theorem

Let $W = \left\{(x_1, x_2, \ldots, x_n) \mid \prod_{i \neq j}(x_i - x_j) \neq 0\right\}$. *Then* $\#(W) \geq n!$

Since $logn! = \Omega(nlogn)$ this proves the lower bound.

Element Distinctness Lower Bounds

Proof.

For each permutation σ let:

$$W_\sigma = \{(x_1, x_2, \ldots, x_n) \mid x_{\sigma(1)} < x_{\sigma(2)} < \cdots < x_{\sigma(n)}\}$$

W_σ is the set of all tuples $(x_1, x_2, \ldots, x_n)$ which respect the order given by σ and $W_\sigma \subseteq W$. It suffices to prove that for all $\sigma' \neq \sigma$ the sets $W_{\sigma'}$ and W_σ are not connected.
For any distinct permutations σ and σ' there exist two distinct $1 \leq i, j \leq n$ such that:

$$\sigma^{-1}(i) < \sigma^{-1}(j) \implies X_j - X_i > 0$$
$$\sigma'^{-1}(i) > \sigma'^{-1}(j) \implies X_j - X_i < 0$$

Element Distinctness Lower Bounds

Proof.

These two points belong to W_σ and $W_{\sigma'}$ respectively. Lets try find a path from one to the other. This path must pass from a point z that make the term $X_i - X_j = 0$. Since both are subsets of W this would violate $X_i - X_j \neq 0$. So this point cannot belong to either W_σ or $W_{\sigma'}$ and these are not connected.
Thus all distinct permutations are not connected and $\#(W) \geq n!$. $\square$

From Ben-Or's theorem this implies that the lower bound for `Element Distinctness` is $\Omega(nlogn)$.

- S.Arora, B.Barak, "Computational Complexity, A Modern Approach"
- M.Sudan, Lectures notes on Algebra and Computation