6.966 Algebra and Computation	December 7, 1998

Lecture 25

Lecturer: Madhu Sudan — *Scribe: Amit Sahai*

1 Introduction

In this lecture, we introduce Mulmuley's Algebraic Parallel computation model (Algebraic PRAM), and give a very high level overview of Mulmuley's Lower Bound techniques in this model without bit operations. For more details, see Mulmuley's paper "Lower Bounds in a Parallel Model without bit operations," available at `http://www.cs.uchicago.edu/~mulmuley/`.

2 Algebraic PRAM Model

Mulmuley introduces the Algebraic PRAM model, which we will consider in the non-uniform setting. We think of each of p processors as being defined by a separate algebraic computation tree, each with some specified topological ordering indicating the sequence in which nodes are to be evaluated. We allow the intermediate variables of any processor at time t to be used by all other processors at time $t+1$. We think of all operations as being done over the integers; thus rational computations are possible since they can be simulated by integer operations on a pair of integers corresponding to the numerator and denominator of a rational number. We allow the computation to use the standard algebraic operations, $+, -, *$ and branching operations based on ≥ 0 (and hence also $= 0$). Note that all these operations and $/$ can be simulated for rationals with only a constant factor overhead. For simplicity in certain arguments, we will also consider the *Linear* PRAM model, in which we require that one of the operands in any $*$ operation be a constant.

The main novelty of the model is that we allow the running time and number of processors to depend not only on the number of integers n in the input, but also the *bit length* N of the input. On the other hand, we only charge a unit cost for every algebraic operation, regardless of the bit lengths of the operands.

The main restriction of this model is that bit operations are not allowed. Note, however, that the i'th bit can be extracted in $O(N)$ time by repeated $*, -$ and ≥ 0 operations. It is not clear how this can be significantly speeded up in the parallel setting, however. Indeed, we will show an effective lower bound for this problem later on.

3 Lower Bound Results of Mulmuley

Mulmuley shows that several natural problems in P, in particular MAX-FLOW, MINCOST FLOW, and LINEAR PROGRAMMING, cannot be solved in time $c\sqrt{n}$ with $2^{c\sqrt{n}}$ processors for some positive constant c, where n denotes either the number of nodes in the underlying graph or the total number of variables and constraints. In particular, this gives a separation of Algebraic-NC from P. Note NC is the class of languages that can be recognized by machines with a polynomial number of processors running in polylogarithmic time.

A key idea in Mulmuley's work is to look at a decision problem restricted to a very low dimensional space, and apply algebraic lower-bound techniques on the resulting problem.

4 Intuition: Lower Bounds for $\lfloor x \rfloor$

Is there hope in applying algebraic lower-bound techniques in very small dimensions? We begin by looking at one place we can apply the connected-components technique from the last lecture on a very low-dimensional problem. We consider the problem of determining whether some number $y \geq \lfloor x \rfloor$. This problem is clearly very closely related to the bit representation of x so we expect it to be hard. However, the YES set has only one connected component. We can easily get around this problem by adding the additional constraint that $y < x$.

More formally, let x and y be given rational numbers (each given by a pair of integers). We will consider YES instances to be those where $y < x$ but $y \geq \lfloor x \rfloor$, as shown in Figure 1 below. The first thing to note is that the YES set has infinitely many connected components, so in the usual computation model over the reals (not the one considered here, where time is measured in terms of the bit size of the inputs), by the theorem proved in the last lecture, this problem is undecidable. Of course, this should not surprise anyone, since this problem is essentially a problem about bits, so its undecidability in a model without bit operations is much like the impossibility of computing the NOT function with a monotone circuit.

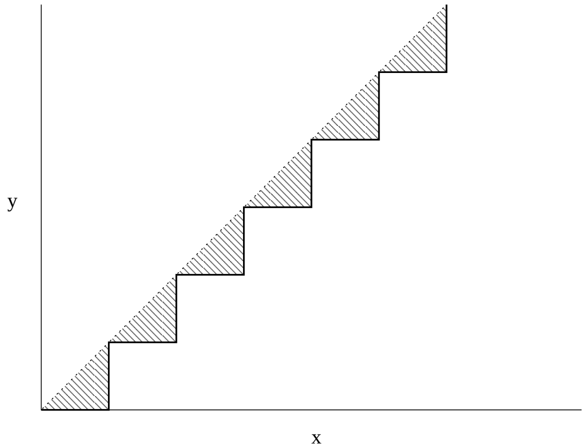

Figure 1.

Remembering that we have a bound N on the number of bits needed to represent x and y, we see that this implies that $|x| \leq 2^N$ and $|y| \leq 2^N$. Hence, the number of connected components of the YES set is $2^{\Omega(N)}$, so it seems that we arrive at an $\Omega(n)$ lower bound for the *sequential* time complexity of this problem. Even this, however, is not quite correct, as the lower bound theorem from the last lecture only applies to computations with real numbers, not rationals with bounded precision. Indeed, it could be that the restriction of a computational problem to bounded precision inputs becomes much simpler. The techniques used by Mulmuley to deal with this problem of finite precision inputs are technically very involved, and we will give only the simplest intuition as to how this problem is dealt with at the end of the lecture.

For now, let us assume these types of arguments can be made to work in the bounded precision model. The challenge is to now convert this sequential lower bound into one for *parallel* computation. We will show how to achieve the bound that this problem cannot be solved in time $c\sqrt{N}$ with $2^{c\sqrt{N}}$ processors for some positive constant c.

5 Parametric Complexity and MAX-FLOW

We would also like to generalize arguments like the one above to work for more natural problems. As one might suspect, however, we cannot hope for low-dimensional natural problems to break into many connected components so easily. Instead, we will focus on problems where the surface separating YES instances from NO instances is piecewise-linear, and consider the number of *slope changes* in this surface. Mulmuley is able to show that a lower bound based on the number of slope changes holds similar to the one known based on the number of connected components. We will only be able to give high-level intuition for why such an argument might work in this lecture. In the technical part of this lecture, we will instead concentrate on showing that a large number of connected components imply lower bounds on parallel algebraic complexity. Now, we will see how natural problems can yield low-dimensional problems with many slope changes. We consider, for example, the problem of MAX-FLOW:

MAX-FLOW: Given a graph $G = (V, E)$ on n vertices, with distinguished vertices s and t, rational capacities $C = \{c_e | e \in E\}$, and a rational number f, decide if there exists a flow from s to t in G of value at least f.

This is a problem of high dimension, so the first thing we do is restrict to a low dimensional version of the problem. For a fixed such graph $G = (V, E)$ and sets of rational numbers $A = \{a_e | e \in E\}$ and $B = \{b_e | e \in E\}$, we define the problem:

PARAMETRIZED-MAX-FLOW: Given a rational number λ and rational number f, decide if there exists a flow of value at least f, in the graph G with capacities $C(\lambda) = \{c_e = a_e + \lambda b_e\}$.

We will denote by $F(\lambda)$ the maximum flow value in the graph G with capacities $C(\lambda)$. These types of problems have been studied in the context of "Parametric Complexity," and it is interesting that a connection exists between that area and Algebraic Complexity. In particular, for this problem, it is known that there exist graphs G on n vertices and sets of rationals $A = \{a_e | e \in E\}$ and $B = \{b_e | e \in E\}$, such that each rational in the sets A and B can be specified with $O(n)$ bits, and such that the function $F(\lambda)$ is piecewise linear with $2^{\Omega n}$ slope changes, all occuring at values of λ specified by $O(n^2)$ bits.

6 The Lower Bound

To summarize, there are three mountains we must climb in order to establish lower bounds for parallel computation with restricted bit lengths. These are:

1. To translate the known argument for lower bounds for sequential computation to one for parallel computation. In particular, we must show that p processors and 1 unit of time do not give as much power as 1 processor with p units of time.
2. To adapt the lower bound techniques for computations with infinite-precision inputs to ones for inputs of bounded precision.
3. To adapt the lower bound techniques to work with the complexity measure of the number of slope changes in a piecewise-linear convex function, rather than the number of connected components.

We will concentrate on showing how to do Item 1. We will give some intuition for Item 2 at the end of the lecture. As for Item 3, we give only the following high-level intuition:

When we were considering the problem of determining whether $y \geq \lfloor x \rfloor$ – a piecewise constant constraint with many discontinuities – initially we had only one connected component. We got around this problem by adding the simple linear constraint $y < x$. Similarly for piecewise linear constraints with many slope changes, we could imagine adding constraints that would break it up into many connected components.

6.1 Linear PRAM

Here, we will consider establishing a lower bound in the Linear PRAM model, where the PRAM can only make linear computations of its inputs. Thus, each branching in the computation divides the space of possible inputs into two linear halfspaces.

With a single processor, after t time steps, it is easy to see that the space of possible inputs can be divided into 2^t regions, as shown pictorially below in Figure 2.

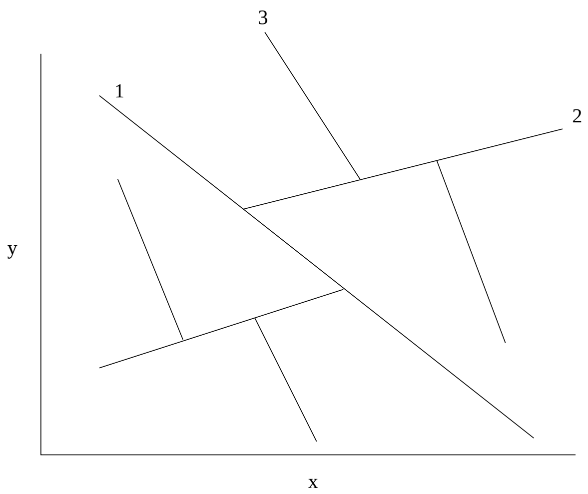

Figure 2. Decomposition after 3 time steps.

But with p processors and only 1 time step, the picture is very different; the half-spaces must be given all at once, as shown in Figure 3.

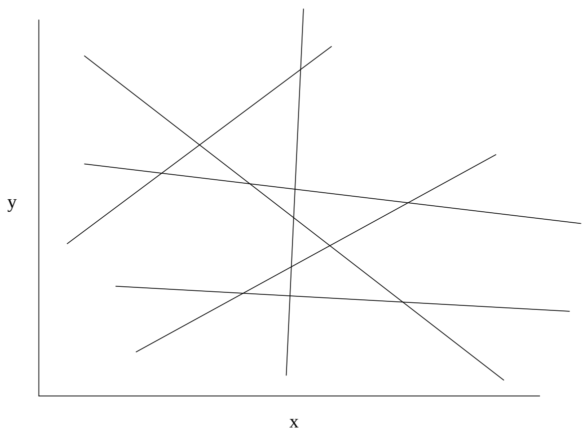

Figure 3. Decomposition by a parallel compuatation in one time step.

It is easy to see that p half-spaces in d dimensions can decompose $\mathbb{R}^d$ into at most $O(p^d)$ cells. Thus, if the dimension is small, this is much weaker than what can be done sequentially with p time steps. This allows us to prove in a straightforward manner the following Lemma:

Lemma 1 *A Linear PRAM computation with p processors and t time steps in d-dimensional space can partition the space into at most $O(p^{dt})$ cells.*

Proof We will proceed by induction on t. The base case was already argued. Consider a cell at time $t-1$. At time t, the p processors can specify up to p half-spaces. This can partition the cell into at most $O(p^d)$ sub-cells. Hence, using the induction hypothesis, after t time steps, we end up with at most $O(p^{d(t-1)}) \cdot O(p^d) = O(p^{dt})$ cells. ■

It simple to see that this Lemma implies the desired lower bound on p and t.

6.2 Algebraic PRAM

For the general algebraic case, a key ingredient will be a version of a theorem of Milnor and Thom:

Theorem 2 (Milnor-Thom) *Given p degree D polynomials $g_1, \ldots, g_p$ in d variables, the number of connected components of $\{v \in \mathbb{R}^d \mid g_1(v) \geq 0, \ldots, g_p(v) \geq 0\}$ is bounded by $(p \cdot D)^{O(d)}$.*

We proceed with the same inductive argument as before, but now at time t, Processor i can branch based on a degree 2^{t-1} inequality g_i. We need to bound the number of connected components in the partition induced by these inequalities. Note that this is *not* just the number of connected components of $\{v \in \mathbb{R}^d \mid g_1(v) \geq 0, \ldots, g_p(v) \geq 0\}$, which can be bounded directly by the Milnor-Thom Theorem. We have to find the number of connected components induced by *all* the possible branches at time t (for example when $g_i(v) < 0$).

Here, we must recall that our bounds are for the case that the inputs are specified by rational points of bounded precision. Hence, they must come from some sufficiently fine lattice. If none of these lattice points v are such that $g_i(v) = 0$ for some i, then the number of connected components *that contain a lattice point* induced by the inequalities can be bounded by applying the Milnor-Thom Theorem to $\{v \in \mathbb{R}^d \mid g_1^2(v) > 0, \ldots, g_p^2(v) > 0\}$. If however, some lattice point v is such that $g_i(v) = 0$, we can (for the sake of argument) replace g_i with $f_i = g_i - \epsilon$ and $h_i = g_i + \epsilon$, for some ϵ chosen small enough so that neither f_i nor h_i contains any lattice point, and so that all the lattice points v such that $g_i(v) > 0$ are now such that $f_i(v) > 0$, while all lattice points v such that $g_i(v) < 0$ are now such that $h_i(v) < 0$. (See Figures 4 and5 below.) Now we may apply the Milnor-Thom Theorem to the set $\{v \in \mathbb{R}^d \mid f_1^2(v) > 0, \ldots, f_p^2(v) > 0, h_1^2(v) > 0, \ldots, h_p^2(v) > 0, \}$, to arrive at a bound of at most $(4p \cdot 2^{t-1})^{O(d)}$ subcells created per cell at time t.

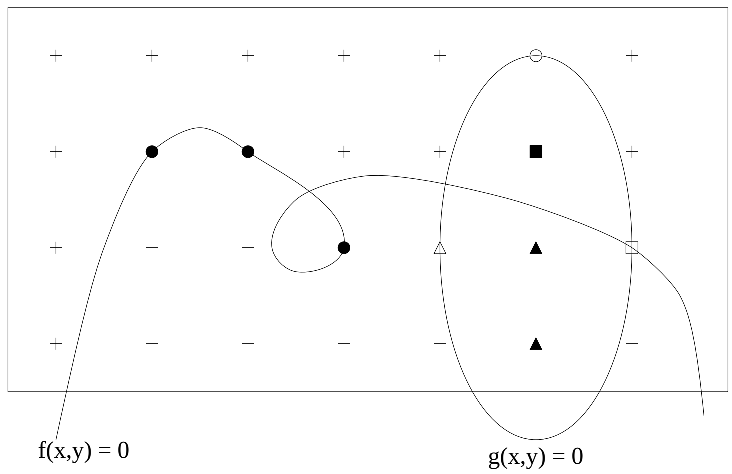

Figure 4. Two algebraic curves and lattice points.

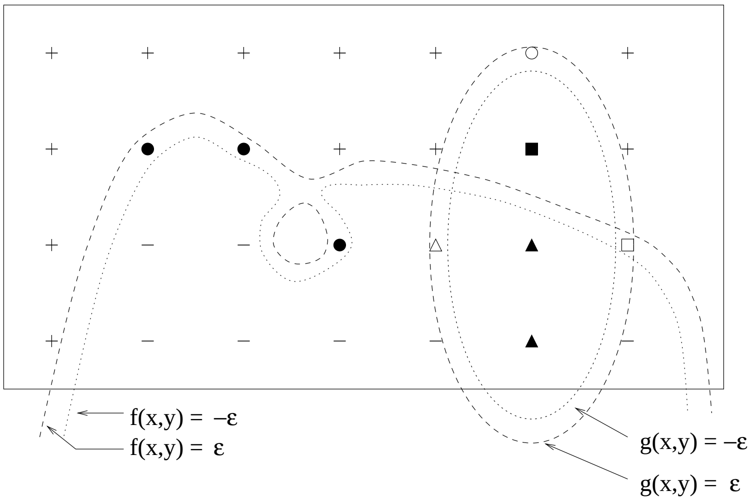

Figure 5. ϵ-shifting of two algebraic curves.

This shows that at most $O(p^d \cdot 2^{td})^t = O(p^{dt} \cdot 2^{t^2 d})$ cells can exist at time t in a general Algebraic PRAM computation.

Thus, for constant d, there exists some constant c such that if $p \leq 2^{c\sqrt{n}}$ and $t \leq c\sqrt{n}$, then at most 2^n cells can exist after t timesteps for an Algebraic PRAM with p processors. This gives the lower bound we were after.

7 Intuition for bounded precision

In this final section, we give some high-level intuition of how the proof proceeds to deal with the problem of bounded precision. We must essentially show that the set of rational points that can be generated in our bounded precision model is sufficiently dense to force many cells to contain a point from our set for any cell decomposition that could come from a feasible computation.

Recall that in our model, rational numbers were actually specified by a pair of bounded integers. We were also considering some problems (like the $\lfloor x \rfloor$ problem or the parametrized version of MAX-FLOW) which involved two rational numbers. We can consider these problems as problems that take three bounded integers w, y, z and use w/z as the first rational and y/z as the second. In general we are trying to lower bound the complexity of deciding a question like is $w/z \leq C(y/z)$ for some piecewise-linear function C. As shown in Figure 6 below, we can consider the possible rational points in our two variable problem to be the projections of some large cube of the 3-dimensional integer lattice onto the plane where $z = 1$. Mulmuley argues that if the bounds on w, y, and z are reasonably large (like $1 \leq w, y, z \leq 2^{n^2}$), then the density of the projection onto $z = 1$ is high enough to establish the result.

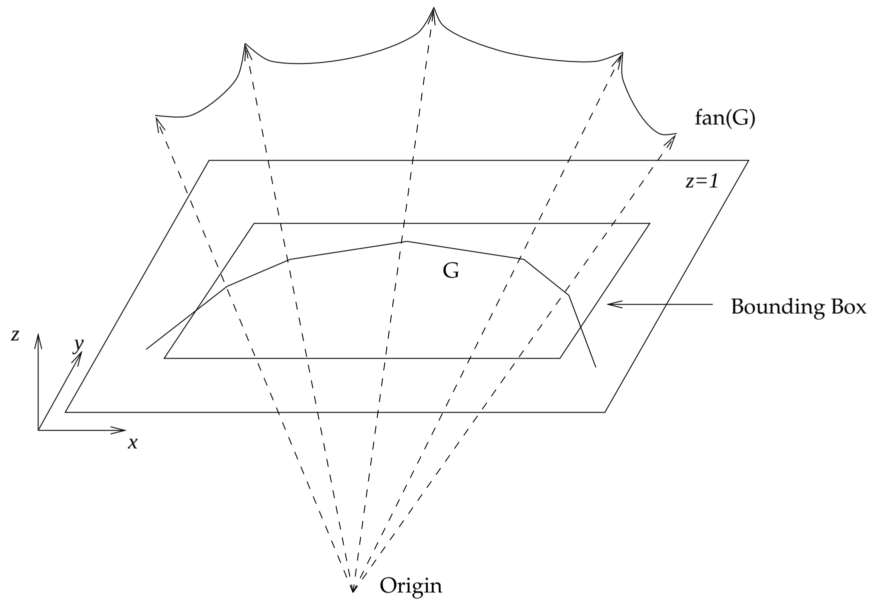

Figure 6. Projecting the integer lattice onto $z = 1$.