2.3 Lecture 10 (24/Oct): Perfect matchings, polynomial identity testing

2.3.1 Matchings

A *matching* in a graph $G = (V, E)$ is a set of vertex disjoint edges; the size of the matching is the number of edges.

Let $n = |V|$ and $m = |E|$. A *perfect matching* is one of size $n/2$. A *maximal matching* is one to which no edges can be added. A *maximum matching* is one of largest size.

How hard are the problems of finding such objects?

It is of course easy to find a maximal matching—sequentially. On the other hand, finding one on a parallel computer is a much more interesting problem, which I hope to return to later in the course.

Returning to sequential computation: Finding a maximum matching, or deciding whether a perfect matching exists, are interesting problems. In bipartite graphs, Hall's theorem and the augmenting path method give very nice and accessible deterministic algorithms for maximum matching. In general graphs the problem is harder but there are deterministic algorithms running in time $O(\sqrt{n}m)$ [75, 43].

2.3.2 Bipartite perfect matching: deciding existence

The first problem we focus on here is to decide whether a bipartite graph has a perfect matching. As noted there are nice deterministic algorithms for this problem but the randomized one is even simpler. Write $G = (V_1, V_2, E)$ with $E \subseteq V_1 \times V_2$. Form the $V_1 \times V_2$ "variable" matrix A which has $A_{ij} = x_{ij}$ if $\{i, j\} \in E$, and otherwise $A_{ij} = 0$.

Let q be some prime power and consider the x_{ij} as variables in $GF(q)$. The determinant of A, then, is a polynomial in the variables x_{ij}.

Before launching into this, a word on a subtle point: what does it mean for a (multivariate) polynomial p to be nonzero? Consider a polynomial over any field κ, which is to say, the coefficients of all the monomials in the polynomial lie in κ.

Definition 31 *We consider a polynomial nonzero if any monomial has a nonzero coefficient.*

A *stronger* condition, which is *not* the definition we adopt, is that $p(x) \neq 0$ for some $x \in \kappa$. Of course this implies the condition in the definition; but it is strictly stronger, as we can see from the example of the polynomial $x^2 + x$ over the field $\mathbb{Z}/2$.

However, the conditions are equivalent in the following two cases:

1. Over infinite fields such as $\mathbb{R}$ or $\mathbb{C}$.

   This will follow from Lemma 37.

2. For multilinear polynomials. (This applies in particular to $\mathrm{Det}(A)$ which we are considering now, so for Lemma 33, it wouldn't have mattered which definition we used.) Specifically we have:

   Lemma 32 *Let $p(\vec{x})$ be a nonzero multilinear polynomial over field κ. Then there is a setting of the x_i to values c_i in κ s.t. $p(\vec{c}) \neq 0$.*

Proof: Every monomial is associated with a set of variables; choose a minimal such set. (E.g., if there is a constant term, then the empty set.) Assign the value 1 to all variables in this set, and 0 to all variables outside this set. Only the chosen monomial can be nonzero, so $p \neq 0$ for this assignment. □

Lemma 33 *G has a perfect matching iff* $\mathrm{Det}(A) \neq 0$.

(This is the "baby" version of a result of Tutte that we will see later in Theorem 38.)

Proof: Every monomial in the expansion of the determinant corresponds to a permutation. A permutation is simply a pattern of 1's hitting each edge and column exactly once, namely, a perfect matching in the bipartite graph. Conversely, if some perfect matching is present, it puts a monomial in the determinant with coefficient either 1 or -1. □

Corollary 34 *Fix any field κ. G has a perfect matching iff there is an assignment of the variables in A such that the determinant is nonzero.*

Proof: Apply Lemma 32 to Lemma 33. □

This suggests the following exceptionally simple algorithm: compute the polynomial and see if it is nonzero.

There's a problem with this idea! The determinant has exponentially many monomials. This is not a problem for computing determinants over a ring such as the integers, because even the sum of exponentially many integers only has polynomially more bits than the largest of those integers has. However, in this ring of multivariate polynomials (i.e., the ring $\kappa[\{x_{ij}\}]$ where $\kappa = \mathbb{Q}$ or $\kappa = GF(q)$, for the moment it doesn't matter), there are exponentially many *distinct* terms to keep track of if you want to write the polynomial out as a sum of monomials. Of course the determinant has a more concise representation (namely, as "$\mathrm{Det}(A)$"), but we do not know how to efficiently convert that to any representation that displays transparently whether the polynomial is the 0 polynomial.

So we modify the original suggestion. Since we do know how to efficiently compute determinants of scalar matrices, let's substitute scalar values for the x_{ij}'s. What values should we use? Random ones.

Revised Algorithm: Sample the x_{ij}'s u.a.r. in $GF(q)$; call the sampled matrix A_R. Compute $\mathrm{Det}(A_R)$; report "G has/hasn't a perfect matching" according to whether $\mathrm{Det}(A_R) \neq 0$ or $= 0$.

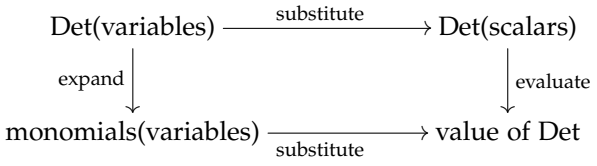

Figure 2.1: This diagram commutes, but for a fast commute, go right and then down.

Clearly the algorithm answers correctly if there is no perfect matching, and it is fast (see Fig. 2.3.2). What needs to be shown is that the probability of error is small if there is a perfect matching (and q is large enough). So this is an RP-type algorithm for "G has a perfect matching".

Theorem 35 *The algorithm is error-free on bipartite graphs lacking a perfect matching, and the probability of error of the algorithm on bipartite graphs which have a perfect matching is at most n/q. The runtime of the algorithm is $n^{\omega+o(1)}$, where ω is the matrix multiplication exponent.*

All we have to do, then, is use a prime power $q \geq 2n$ in order to have error probability $\leq 1/2$.

Incidentally, there is always a prime $2n \leq q < 4n$; this is called "Bertram's postulate". This fact alone isn't quite strong enough for if we want to find a prime in the right size range efficiently, but that too can be done, in a slightly larger range. (The density of primes of this size is about $1/\log n$ so we don't have to try many values before we should get lucky; and note, primality testing has efficient algorithms in ZPP and even somewhat less efficient algorithms in P [4].) However, we don't have to work this hard, since we're satisfied with prime powers rather than primes. We can simply use the first power of 2 after $2n$.

We will prove Theorem 35 after introducing a general useful tool.

2.3.3 Polynomial identity testing

In the previous section we saw that testing for existence of a perfect matching in a bipartite graph can be cast as a special case of the following problem. We are given a polynomial $p(x)$, of total degree n, in variables $x = (x_1, \ldots, x_m)$, $m \geq 1$. (The total degree of a monomial is the sum of the degrees of the variables in it; the total degree of a polynomial is the greatest total degree of its monomials.) We are agnostic as to how we are "given" the polynomial, and demand only that we be able to quickly evaluate it at any scalar assignment to the variables. We wish to test whether the polynomial is identically 0, and our procedure for doing so is to evaluate it at a random point and report "yes" if the value there is 0. We rely on the following lemma. Let $z(p)$ be the set of roots (zeros) of a polynomial p.

Lemma 36 *Let p be a nonzero polynomial over $GF(q)$, of total degree n in m variables. Then $|z(p)| \leq nq^{m-1}$.*

As a fraction, this is saying that $|z(p)|/q^m \leq n/q$, and in this form the lemma immediately implies Theorem 35.

The univariate case of the lemma is probably familiar to you.

The lemma is a special case of the following more general statement which holds for any, even infinite, field κ.

Lemma 37 *Let p be a nonzero polynomial over a field κ, of total degree n in variables $x_1, \ldots, x_m$. Let $S_1, \ldots, S_m$ be subsets of κ with $|S_i| \leq s$ for all i. Then $|z(p) \cap (S_1 \times \ldots \times S_m)| \leq s^{m-1}n$.*

This is usually known as the Schwartz-Zippel lemma [90, 105], although the results in these two publications were not precisely equivalent, and there were at least two other discoveries of versions of the result, by Ore [70] and by DeMillo and Lipton [28]. A generalization beyond polynomials is due to Gonnet [46].

Recalling the two candidate definitions of what it means for a polynomial to be nonzero, since in Defn 31 we chose the weaker condition, Lemma 37 is stronger than it would be otherwise.

Proof: of Lemma 37: The lemma is trivial if $n \geq s$, so suppose $n < s$.

First consider the univariate case, $m = 1$. (In this case the two lemmas are identical since any set S_1 is a product set.) This follows by induction on n because if $n \geq 1$ and $p(\alpha) = 0$, then p can be factored as $p(x) = (x - \alpha) \cdot q(x)$ for some q of degree $n - 1$. (Because, make the change of variables to $x - \alpha$. After this change the polynomial cannot have any constant term. So we can factor out $(x - \alpha)$.)

Next we handle $m > 1$ by induction. If x_1 does not appear in p then the conclusion follows from the case $m-1$. Otherwise, write p in the form $p(x) = \sum_0^n x_1^i p_i(x_2, \ldots, x_m)$, and let $0 < i \leq n$ be largest such that $p_i \neq 0$. The degree of p_i is at most $n-i$, so by induction,

$$\frac{|z(p_i) \cap (S_2 \times \ldots \times S_m)|}{s^{m-1}} \leq \frac{n-i}{s}$$

For $(x_2, \ldots, x_m) \in z(p_i)$ we allow as a worst case that all choices of $x_1 \in S_1$ yield a zero of p.

For $(x_2, \ldots, x_m) \notin z(p_i)$, p restricts to a nonzero polynomial of degree i in the variable x_1, so by the case $m = 1$,

$$\frac{|z(p_i) \cap (S_1 \times x_2 \times \ldots \times x_m)|}{s} \leq \frac{i}{s}$$

Since $\frac{i}{s} \leq \frac{n}{s} < 1$, the weighted average of our two bounds (on the fraction of roots in sets of the form $S_1 \times x_2 \times \ldots \times x_m$) is worst when the set $z(p_i) \cap (S_2 \times \ldots \times S_m)$ is as large as possible. Thus

$$\frac{z(p) \cap (S_1 \times \ldots \times S_m)}{s^m} \leq \frac{n-i}{s} \cdot 1 + (1 - \frac{n-i}{s}) \cdot \frac{i}{s} = \frac{n}{s} - \frac{i(n-i)}{s} \leq \frac{n}{s}$$

□

Comment: This lemma gives us an efficient randomized way of testing whether a polynomial is identically zero, and naturally, people have wondered whether there might be an efficient deterministic algorithm for the same task. So far, no such algorithm has been found, and it is known that any such algorithm would have hardness implications in complexity theory that are currently out of reach [58][1].

[1]Specifically: If one can test in polynomial time whether a given arithmetic circuit over the integers computes the zero polynomial, then either (i) NEXP $\not\subseteq$ P/poly or (ii) the Permanent is not computable by polynomial-size arithmetic circuits.