4 Lecture 4, October 8, 2014

4.1 Graph Crossing Number

A graph G on n vertices and m edges can always be drawn in the plane with edges represented by smooth arcs, crossing only possibly at finitely many points (and by definition including their endpoints). We'll call the least number of such crossings, $c(G)$, the crossing number of the graph. If $c(G) = 0$, the graph is called planar. If a planar graph is drawn without crossings, it leaves some f connected components in its complement (the complement of the union of the arcs).

Euler's formula states that in a planar drawing of a connected graph,

$$n - m + f = 2 \tag{10}$$

The proof is simple: start with a spanning tree. Adding edges simultaneously increments m and f.

Now, consider that every face is adjacent to at least three edges. Since each edge is adjacent to exactly two faces, we have $3f \leq 2m$. Substituting into (10) yields $n - m/3 \geq 2$ or

$$m \leq 3n - 6 \tag{11}$$

Our purpose now is to show that a dense graph must have high crossing number. We will start with a weak bound. Then in a remarkable application of the probabilistic method, we will be able to amplify the weak bound to a stronger one—without any more geometric reasoning whatsoever.

Theorem 15 $c(G) \geq m - 3n + 6$

This weak bound is already sufficient to show that K_5, the complete graph on 5 vertices, is not planar.

Proof: Given an optimal planar drawing, replace each crossing by a new vertex. This has the effect of increasing n by $c(G)$ and m by $2c(G)$, and creates a planar graph. So applying Eqn (11) we have $m + 2c(G) \leq 3(n + c(G)) - 6$. □

Now we show the probabilistic amplification.

Theorem 16 $c(G) \geq 4m^3/243n^2$.

This is tight up to the constant. The argument we show is due to Chazelle, Sharir and Welzl, although the theorem (with some constant) predates their argument [37, 4].

The best known lower bound is $m^3/33.75n^2 - 0.9n$ [44]; the same paper supplies an upper bound that is matching up to constants: $c(G) \leq 0.06(1 + o(1))m^3/n^2$. (The example is quite simple: points in a $\sqrt{n} \times \sqrt{n}$ grid are connected if within a certain distance.)

Proof: The idea is to fix an optimal planar drawing of G with $c(G)$ crossings, and then to examine a random subgraph G' of G, in which each vertex is retained independently with probability p. We let n' be the number of vertices of G which are retained in G'; let m' be the number of edges of G which are retained in G'; and let c' be the number of crossings in the drawing of G which are retained, as crossings, in the same drawing of G'. Then, applying Theorem (15), we have $c' \geq c(G') \geq m' - 3n' + 6$. Consequently,

$$E(c') \geq E(m') - 3n' + 6$$

It is straightforward to note that $E(n') = pn$. Since G has no self-loops, it is also straightforward to note that $E(m') = p^2 m$. In order to evaluate $E(c')$, we note the following three properties of an optimal planar drawing of G, all of which can be shown by simple redrawings of the graph:

1. No edge crosses itself.
2. No two edges cross more than once.

3. No two edges that share a vertex cross.

Applying these to the drawing of G, we conclude in particular that every crossing in the drawing involves two distinct edges of G, emanating from four distinct vertices, and therefore $E(c') = p^4 c(G)$. Thus we have

$$p^4 c(G) \geq p^2 m - 3pn + 6$$

which implies

$$c(G) \geq m/p^2 - 3n/p^3$$

Optimizing by setting $p = 9n/2m$ we find

$$c(G) \geq \frac{4}{243}\frac{m^3}{n^2}.$$

□

4.2 Algebraic Fingerprinting: matrix multiplication, associativity, matchings, polynomial identity testing

There are several key ways in which randomness is used in algorithms. One is to "push apart" things that are different even if they are similar. We'll study a few examples of this phenomenon.

4.3 Verifying Matrix Multiplication

It is a familiar theme that *verifying* a fact may be easier than *computing* it. Most famously, it is widely conjectured that P≠NP. Now we shall see a more down-to-earth example of this phenomenon.

In what follows, all matrices are $n \times n$. In order to eliminate some technical issues (mainly numerical precision, also the design of a substitute for uniform sampling), we suppose that the entries of the matrices lie in $\mathbb{Z}/p$, p prime; and that scalar arithmetic can be performed in unit time.

(The same method will work for any finite field and a similar method will work if the entries are integers less than $\text{poly}(n)$ in absolute value, so that we can again reasonably sweep the runtime for scalar arithmetic under the rug.)

Here are two closely related questions:

1. Given matrices A, B, *compute* $A \cdot B$.
2. Given matrices A, B, C, *verify* whether $C = A \cdot B$.

The best known algorithm for the first of these problems runs in time $O(n^{2.3728639})$ [24]. Resolving the correct exponent is a major question in computer science.

Clearly the second problem is no harder, and a lower bound of $\Omega(n^2)$ even for that is obvious since one must read the whole input.

Randomness is not known to help with problem (1), but the situation for problem (2) is quite different.

Theorem 17 (Freivalds [22]) *There is an RP algorithm for the language "$C = A \cdot B$", running in time $O(n^2)$.*

Definition 18 *BPP, RP, coRP, ZPP: These are the four main classes of randomized polynomial-time computation. All are decision classes. A language L is:*

- *In BPP if the algorithm errs with probability $\leq 1/3$.*

- *In RP if for $x \in L$ the algorithm errs with probability $\leq 1/3$, and for $x \notin L$ the algorithm errs with probability 0.*

while the subsidiary definitions are:

- *$L \in$ coRP if $L^c \in RP$, that is to say, if for $x \notin L$ the algorithm errs with probability $\leq 1/3$, and for $x \in L$ the algorithm errs with probability 0.*
- *ZPP = RP $\cap$ coRP. There are two equivalent ways of thinking about this class: (a) a poly-time randomized algorithm that with probability $\geq 1/3$ outputs the correct answer, and with the remaining probability halts and outputs "don't know"; (b) an expected-poly-time algorithm that always outputs the correct answer.*

It is a routine exercise that none of these constants matter and can be replaced by any $1/poly$, although completing that exercise relies on the Chernoff bound which we'll see in a later lecture.

Proof: Note that the obvious procedure for matrix-vector multiplication runs in time $O(n^2)$.

The verification algorithm is simple. Select uniformly a vector $x \in (\mathbb{Z}/p)^n$. Check whether $ABx = Cx$ without ever multiplying AB: applying associativity, $(AB)x = A(Bx)$, this can be done in just three matrix-vector multiplications. Output "Yes" if the the equality holds; output "No" if it fails. Clearly $AB = C$, the output will be correct. In order to show that the problem is in RP, it remains to show that

$$\Pr(ABx = Cx | AB \neq C) \leq 1/2.$$

The event $ABx = Cx$ is equivalently stated as the event that x lies in the right kernel of $AB - C$. Given that $AB \neq C$, that kernel is a *strict* subspace of $(\mathbb{Z}/p)^n$ and therefore of at most half the cardinality of the larger space. Since we select x uniformly, the probability that it is in the kernel is at most $1/2$. □

4.4 Verifying Associativity

Let a set S of size n be given, along with a binary operation $\circ : S \times S \to S$. Thus the input is a table of size n^2; we call the input $(S, \circ)$. The problem we consider is testing whether the operation is associative, that is, whether for all $a, b, c \in S$,

$$(a \circ b) \circ c = a \circ (b \circ c) \tag{12}$$

A triple for which (12) fails is said to be a nonassociative triple.

No sub-cubic-time deterministic algorithm is known for this problem. However,

Theorem 19 (Rajagopalan & Schulman [46]) *There is an RP algorithm for associativity running in time $O(n^2)$.*

Proof: An obvious idea is to replace the $O(n^3)$-time exhaustive search for a nonassociative triple, by randomly sampling triples and and checking them. The runtime required is inverse to the fraction of nonassociative triples, so this method would improve on exhaustive search if we were guaranteed that a nonassociative operation had a super-constant number of nonassociative triples. However, for every $n \geq 3$ there exist nonassociative operations with only a single nonassociative triple.

So we'll have to do something more interesting.

Let's define a binary operation $(\mathbb{S}, \circ)$ on a much *bigger* set $\mathbb{S}$. Define $\mathbb{S}$ to be the vector space with basis S over the field $\mathbb{Z}/2$, that is to say, an element $x \in \mathbb{S}$ is a formal sum

$$x = \sum_{a \in S} a x_a \quad \text{for } x_a \in \mathbb{Z}/2$$

The product of two such elements x, y is

$$\begin{aligned} x \circ y &= \sum_{a \in S} \sum_{b \in S} (a \circ b) x_a y_b \\ &= \sum_{c \in S} c \bigoplus_{a,b: a \circ b = c} x_a y_b \end{aligned}$$

where of course $\bigoplus$ denotes sum mod 2.

On $(\mathbb{S}, \circ)$ we have an operation that we do not have on $(S, \circ)$, namely, addition:

$$x + y = \sum_{a \in S} a(x_a + y_b)$$

(Those who have seen such constructions before will recognize $(\mathbb{S}, \circ)$ as an "algebra" of $(S, \circ)$ over $\mathbb{Z}/2$.)

The algorithm is now simple: *check the associative identity for three random elements of* $\mathbb{S}$. That is, select x, y, z u.a.r. in $\mathbb{S}$. If $(x \circ y) \circ z = x \circ (y \circ z)$, report that $(S, \circ)$ is associative, otherwise report that it is not associative. The runtime for this process is clearly $O(n^2)$.

If $(S, \circ)$ is associative then clearly so is $(\mathbb{S}, \circ)$, because then $(x \circ y) \circ z)$ and $x \circ (y \circ z)$ have identical expansions as sums. Also, nonnassociativity of $(S, \circ)$ implies nonnassociativity of $(\mathbb{S}, \circ)$ by simply considering "singleton" vectors within the latter.

But this equivalence is not enough. The crux of the argument is the following:

Lemma 20 *If $(S, \circ)$ is nonnassociative then at least one eighth of the triples (x, y, z) in $\mathbb{S}$ are nonassociative triples.*

Proof: The proof relies on a variation on the inclusion-exclusion principle.

For any triple $a, b, c \in S$, let

$$g(a, b, c) = (a \circ b) \circ c - a \circ (b \circ c).$$

Note that g is a mapping $g : S^3 \to \mathbb{S}$. Now extend g to $g : \mathbb{S}^3 \to \mathbb{S}$ by:

$$g(x, y, z) = \sum_{a,b,c} g(a, b, c) x_a y_b z_c$$

If you imagine the $n \times n \times n$ cube indexed by S^3, with each position (a, b, c) filled with the entry $g(a, b, c)$, then $g(x, y, z)$ is the sum of the entries in the combinatorial subcube of positions where $x_a = 1, y_b = 1, z_c = 1$. (We say "combinatorial" only to emphasize that unlike a physical cube, here the slices that participate in the subcube are not in any sense adjacent.)

Fix (a', b', c') to be any nonassociative triple of S.

Partition $\mathbb{S}^3$ into blocks of eight triples apiece, as follows. Each of these blocks is indexed by a triple x, y, z s.t. $x_{a'} = 0, y_{b'} = 0, z_{c'} = 0$. The eight triples are $(x + \varepsilon_1 a', y + \varepsilon_2 b', z + \varepsilon_3 c')$ where $\varepsilon_i \in \{0, 1\}$.

Now observe that

$$\sum_{\varepsilon_1, \varepsilon_2, \varepsilon_3} g(x + \varepsilon_1 a', y + \varepsilon_2 b', z + \varepsilon_3 c') = g(a', b', c')$$

To see this, note that each of the eight terms on the LHS is, as described above, a sum of the entries in a "subcube" of the "S^3 cube". These subcubes are closely related: there is a core subcube whose indicator function is $x \times y \times z$, and all entries of this subcube are summed within all eight terms. Then there are additional width-1 pieces: the entries in the region $a' \times y \times z$ occur in four terms, as do the regions $x \times b' \times z$ and $x \times y \times c'$. The entries in the regions $a' \times b' \times z$, $a' \times y \times c'$ and $x \times b' \times c'$ occur in two terms, and the entry in the region $a' \times b' \times c'$ occurs in one term.

Since the RHS is nonzero, so is at least one of the eight terms on the LHS. □ □