6 Lecture 6, October 13, 2014

6.1 Perfect Matchings in General Graphs: Finding Them. The Isolating Lemma

We now develop a randomized method of Mulmuley, U. Vazirani and V. Vazirani to *find* a perfect matching if one exists. A polynomial time algorithm is implied by the previous testing method along with self-reducibility of the perfect matching decision problem. However, with the following method we can solve the same problem in *parallel*, that is to say, in polylog depth on polynomially many processors. (The deterministic version of this complexity class is known as NC, and the randomized as RNC. More exactly, RNC=$\bigcup_{i\geq 1}$RNC^i where RNC^i allows poly-many processors and depth $O((\log n)^i)$.) This is not actually the first RNC algorithm for this task—that is an RNC^3 method due to [38]—but it is the "most parallel" since it solves the problem in RNC^2.

A slight variant of the method yields a *minimum weight perfect matching* in a weighted graph with "small" weights, that is, integer weights represented in unary; and there is a fairly standard reduction from the problem of finding a *maximum matching* to finding a minimum weight perfect matching in a graph with weights in $\{0,1\}$. So we actually through this method can find a maximum matching in a general graph, with a similar total amount of work.

A key part of the method is the following lemma. First some notation. Let $A = \{a_1, a_2, \ldots, a_m\}$ be a finite set. Let $\mathcal{S} = \{S_1, \ldots, S_k\}$ be a collection of subsets of A. If $a_1, \ldots, a_m$ are assigned weights $w_1, \ldots, w_m$, the weight of set S_i is defined to be $w(S_i) = \sum_{a_j \in S_i} w_j$.

Lemma 27 ([50] Isolating Lemma) *Let the weights $w_1, \ldots, w_m$ be independent random variables, each w_i being sampled uniformly in some set $R_i \subseteq \mathbb{R}$, $|R_i| \geq r$. Then*

$$\Pr[\exists\, i \neq j \text{ s.t. } w(S_i) = w(S_j) = \min_{\ell}\{w(S_\ell)\}] \leq \frac{m}{r} \tag{13}$$

This lemma is remarkable because of the absence of a dependence on k in the conclusion.

Proof: Consider any vector of weights $w_1, \ldots, w_m$ and any index $i \in \{1, \ldots, m\}$. Let

$$\begin{aligned} \alpha_{i,0} &= \min_{j: a_i \notin S_j} w(S_j) \\ \alpha_{i,1} &= \min_{j: a_i \notin S_j} w(S_j - \{a_i\}) \end{aligned}$$

Neither of these quantities depends on w_i. Define the "bad event" B_i to be the event that

$$\alpha_{i,1} + w_i = \alpha_{0,i}. \tag{14}$$

If none of the bad events occur, then there is only a single minimum weight subset, because the direction of the inequality in Eqn. 14 for each i shows whether a_i is in any minimum weight subset.

The bad events may be highly correlated, but no matter. Each B_i occurs with probability at most $1/r$, as we see by first conditioning on the weights other than w_i and then noting that equality in Eqn. 14 can occur only for at most one of the values in R_i. The lemma follows by a union bound. □

Now we describe the algorithm to find a perfect matching (or report that probably none exists) in a graph $G = (V, E)$ with $n = |V|, m = |E|$.

For every $(i, j) \in E$ pick an integer weight w_{ij} iid uniformly distributed in $\{1, \ldots, 2m\}$. By the isolating lemma, there is with probability at least $1/2$ a unique minimum weight perfect matching of G.

Define the matrix T by:

$$T_{ij} = \begin{cases} 0 & \text{if } \{i,j\} \notin E \\ 2^{w_{ij}} & \text{if } \{i,j\} \in E, i < j \\ -2^{w_{ji}} & \text{if } \{i,j\} \in E, i > j \end{cases} \tag{15}$$

This is an instantiation of the Tutte matrix, with $x_{ij} = 2^{w_{ij}}$.

Claim 28 *If there is a unique minimum weight perfect matching of G (call it M) then* $\mathrm{Det}(T) \neq 0$ *and moreover, the highest power of 2 that divides* $\det(T)$ *is* 2^{2W}*, where W is the weight of M. I.e.* $\mathrm{Det}(T) = 2^{2W} \times$ *[an odd number].*

Proof: of Claim: As before we look at the contributions to $\mathrm{Det}(T)$ of all the permutations π that are supported by edges of the graph. The contributions from permutations having odd cycles cancel out—that is just because this is a special case of a Tutte matrix.

It remains to consider permutations π that have only even cycles.

- If π consists of transpositions along the edges of M then it contributes $\pm 2^{2W}$.
- If π has only even cycles, but does not correspond to M, then:
  - If π is some other matching of weight $W' > W$ then it contributes $\pm 2^{2W'}$.
  - If π has only even cycles and at least one of them is of length ≥ 4, then by separating each cycle into a pair of matchings on the vertices of that cycle, π is decomposed into two matchings $M_1 \neq M_2$ of weights W_1, W_2, so π contributes $\pm 2^{W_1+W_2}$. Because of the uniqueness of M not both of M_1 and M_2 can achieve weight W, so $W_1 + W_2 > 2W$. □

Now let

$$\begin{aligned} m_{ij} &= \sum_{\pi:\pi(i)=j} \mathrm{sign}(\pi) \prod_{k=1}^{n} T_{k,\pi(k)} \\ &= \pm 2^{w_{ij}} \, \mathrm{Det}(\hat{T}_{ij}) \end{aligned} \tag{16}$$

where $\hat{T}_{ij}$ is the (i,j)-deleted minor of T (the matrix obtained by removing the i'th row and j'th column from T).

Claim 29 *For every* $\{i, j\} \in E$*:*

1. *The total contribution to* m_{ij} *of permutations* π *having odd cycles is* 0.
2. *If there is a unique minimum weight perfect matching M, then:*
   (a) *If* $\{i, j\} \in M$ *then* $m_{ij}/2^{2W}$ *is odd.*
   (b) *If* $\{i, j\} \notin M$ *then* $m_{ij}/2^{2W}$ *is even.*

Proof: of Claim: This is much like our argument for $\mathrm{Det}(T)$ but localized.

1. If π has an odd cycle then it has an even number of odd cycles and hence an odd cycle not containing point i. Pick the "first" odd cycle that does not contain point i and flip it to obtain a permutation π^r. Note that $(\pi^r)^r = \pi$. The contribution of π^r to m_{ij} is the negation of the contribution of π to m_{ij}, because we have replaced an odd number of terms from the Tutte matrix by the same entry with a flipped sign.
2. By the preceding argument, whether or not $\{i, j\} \in M$, we need only consider permutations containing solely even cycles. Just as argued for Claim 28, the contribution of every such permutation π can be written as $2^{w(M_1)+w(M_2)}$, where M_1 and M_2 are two perfect matchings obtained as follows: each transposition (i, j) in π puts the edge $\{i, j\}$ into both of the matchings; each even cycle of length ≥ 4 can be broken alternatingly into two matchings, one of which (arbitrarily) is put into M_1 and one into M_2.

   The only case in which there is a term for which $w(M_1) + w(M_2) = 2W$ is the single case that $\{i, j\} \in M$ and π consists entirely of transpositions along the edges of M. In every other case, at least one of M_1 or M_2 is distinct from M, and therefore $w(M_1) + w(M_2) > 2W$. The claim follows. □

Finally we collect all the elements necessary to describe the algorithm:

1. Generate the weights w_i uniformly in $\{1, \ldots, 2m\}$.
2. Define T as in Eqn (15), compute its determinant and if it is nonsingular invert it. (Otherwise, start over.) This determinant computation and the inversion can be done (deterministically) in depth $O(\log^2 n)$ by Csanky's algorithm [14]. But a more efficient way is using Pan's randomized algorithm [54], which works in depth $O(\log^2 n)$ and uses $O(n^{3.5}m)$ processors to invert an $n \times n$ matrix with m-bit integers.
3. Determine W by factoring the greatest power of 2 out of $\text{Det}(T)$.
4. Obtain the values $\pm m_{ij}$ from the equations $m_{ij} = \pm 2^{w_{ij}} \text{Det}(\hat{T}_{ij})$ and $\text{Det}(\hat{T}_{ij}) = (-1)^{i+j}(T^{-1})_{ji} \text{Det}(T)$. (Cramer's rule.) If $m_{ij}/2^{2W}$ is odd then place $\{i, j\}$ in the matching.
5. Check whether this defines a perfect matching. This is guaranteed if the minimum weight perfect matching is unique. If a perfect matching was not obtained (which will occur for sure if there is no perfect matching, and with probability $\leq 1/2$ if there is one), generate new weights and repeat the process.

Of course, if the graph has a perfect matching, the probability of incurring k repetitions wihout success is bounded by 2^{-k}, and the expected number of repetitions until success is at most 2.

The simultaneous computation of all the m_{ij}'s in step 2 is key to the efficiency of this procedure.

The numbers in the matrix A are integers bounded by $\pm 2^{2m}$. As mentioned, Pan's algorithm inverts such a matrix using $O(n^{3.5}m)$ processors.

For the maximum matching problem, we use a simple reduction: use weights for each of the non-edges too, but sample those weights uniformly from $2mn + 1, \ldots, 2mn + 2m$ (rather than $1, \ldots, 2m$ like the graph edges). Then no minimum weight perfect matching will use any of the non-edges. The cost of this reduction is that the integers in the matrix now use $O(mn)$ rather than $O(m)$ bits, so the number of processors used by the maximum matching algorithm is $O(n^{4.5}m)$.

(For detail on parallelized linear algebra algorithms see [43] §2.4 & 2.5.5.)