Randomized Algorithms Georgia Tech CS8113F, Winter 1999
Prof. Leonard Schulman CCB 234, (404) 894-6438, schulman@cc.gatech.edu
Lectures # 16 & 17: Mar 8 & 10 '99. "Perfect Matchings in Graphs"
Scribe: Germán Riaño-Mendoza

1 Introduction

Let G be a graph with edge set E. (Undirected, no multiple edges or loops.) A matching is a set of edges joining disjoint vertices. A perfect matching is one that uses all the vertices.

Theorem 1.1 (Schwartz-Zippel). *Let f be a multivariate polynomial of total degree k in the variables $x_1, \dots, x_m$, over the field $GF(q)$. The fraction of vectors $\vec{x} \in GF(q)^m$ which are roots of f is at most k/q.*

The total degree of a polynomial is the maximum, over its monomials, of the sum of the degrees of the variables appearing in the monomial.

For a proof of this theorem see the scribe notes from the previous offering of this course. Note that in the univariate case this is the fundamental theorem of algebra.

Define the matrix A as:

$$A = \begin{cases} 0 & \text{if } (i,j) \notin E \\ x_{ij} & \text{if } (i,j) \in E, i < j \\ -x_{ij} & \text{if } (i,j) \in E, i > j \end{cases} \tag{1}$$

The determinant of A is a polynomial in n^2 variables. We will see that this determinant is different from 0 if and only if there is a perfect matching in G. That allows us to formulate an algorithm to determine if there is a perfect matching.

Theorem 1.2 (Tutte). $\det(A) \neq 0$ *iff A has a perfect matching.*

Proof. If G has perfect matchings choose one and assign 1 to x_{ij} if (i,j) is in that matching, otherwise assign 0 to x_{ij}. The matrix obtained has det $\in \{1, -1\}$.

Conversely assume $\det(A) \neq 0$, we will show that there is a perfect matching. The net contribution to $\det(A)$ of all permutations having an odd cycle is 0, for the following reason. In each such permutation identify the "least" odd cycle by some fixed criterion, e.g. by ordering the odd cycles by the least label of a vertex they contain. Then flip the direction of the least odd cycle. This map is an involution on the set of permutations. Every permutation containing an odd cycle is mapped to another such permutation, whose contribution to the determinant is opposite in sign.

Therefore there are permutations π supported by A (i.e. s.t. $\pi(i) = j \Rightarrow A_{ij} \neq 0$) having only even cycles. In any even cycle of length greater than 2 we can use every alternate edge to obtain a perfect matching. ■

2 Determining whether G has a perfect matching

The following algorithm is due to Rabin and V. Vazirani:

- Pick a prime number q such that $2n \leq q \leq 4n$. (This can be done deterministically in time poly(n) or randomly, Las Vegas (= error free) in time polylog(n). The existence of such a prime is a theorem known as Bertand's postulate, see [1].)
- For every $(i,j) \in E$, select each x_{ij} uniformly independently in $GF(q)$.
- Define the matrix A as in equation (1).
- Test whether $\det(A) = 0$. (This can be done quickly because A has scalar, rather than variable, entries.)

Theorem 2.1 (RV[4]). *If G does not have a perfect matching then* $\det(A) = 0$, *while if it does then* $\det(A) \neq 0$ *with probability at least* $1/2$.

Proof. The total degree of the determinant is n so this follows from the Schwartz-Zippel lemma. ■

3 Finding a perfect matching

We new develop a randomized method to quickly find a perfect matching if one exists. A polynomial time algorithm is implied by the above testing method, and self-reducibility. However, the following method of Mulmuley, U. Vazirani and V. Vazirani is substantially more efficient. It is based upon:

Lemma 3.1 (MVV[3] Isolating Lemma). *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 $w(S_i) = \sum_{a_j \in S_i} w_j$. Let the weights $w_1, \ldots, w_m$ be iid uniform random variables in the range $\{1, \ldots, n\}$. Then*

$$P[\exists\, i \neq j \text{ s.t. } w(S_i) = w(S_j) = \min_{\ell}\{w(S_\ell)\}] \leq \frac{m}{n} \tag{2}$$

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

$$w_{r,i} = (w_1, \ldots, w_{i-1}, r, w_{i+1}, \ldots, w_m), \tag{3}$$

where $r \in \{1, \ldots, n\}$.

Claim: There exists an integer $0 \leq \alpha(w,i) \leq n+1$ such that: (a) If $r > \alpha(w,i)$ then a_i is not contained in any minimal weight set over the weight vector $w_{r,i}$. (b) If $r < \alpha(w,i)$ then a_i is contained in every minimal set over the weight vector $w_{r,i}$.

Proof of the claim: There are three cases.

1. It may be that there is no r for which a_i is contained in any minimal weight set over $w_{r,i}$. Then $\alpha(w,i) = 0$ is satisfactory.

2. It may be that for every r, a_i is contained in every minimal weight set over $w_{r,i}$. Then $\alpha(w,i) = n+1$ is satisfactory.

3. Otherwise set $\alpha(w,i) = \max\{r : a_i \text{ is contained in some minimal weight subset for } w_{r,i}\}$.

   If $r > w_{\alpha(w,i),i}$ then, directly from the definition of $\alpha(w,i)$, no minimal weight subset over $w_{r,i}$ contains a_i.

   If $r < w_{\alpha(w,i),i}$, then compare the weight of any subset over the weight vectors $w_{\alpha(w,i),i}$ and $w_{r,i}$. If the subset contains a_i, its weight in the latter case is decreased by $\alpha(w,i) - r$; otherwise its weight is unchanged. Since among the minimal weight subsets over $w_{\alpha(w,i),i}$ there is at least one which contains a_i, the minimal weight subsets over $w_{r,i}$ are precisely those which are minimal over $w_{\alpha(w,i),i}$ and contain a_i.

Now choose $(w_1, \ldots, w_m)$ uniformly at random from $1, \ldots, n$. Then

$$P(w_i = \alpha(w,i)) \leq 1/n \tag{4}$$

hence

$$P(\text{There is any } i \text{ for which } w_i = \alpha(w,i)) \leq m/n. \tag{5}$$

If for every i, $\alpha(w,i) \neq w_i$, then every a_i either does or does not belong to all minimal subsets. There is therefore a unique minimal subset. ■

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

Now we describe the algorithm to not only decide whether there is a perfect matching in a graph, but also find one, if such exists. The method is also efficiently parallelizable (it is in the complexity class RNC, consisting of problems solvable with randomization in polylogarithmic time on polynomially many processors), but we will not go into this in this lecture.

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

Define the matrix A by:

$$A = \begin{cases} 0 & \text{if } (i,j) \notin E \\ 2^{w_{ij}} & \text{if } (i,j) \in E, i < j \\ -2^{w_{ij}} & \text{if } (i,j) \in E, i > j \end{cases} \tag{6}$$

Claim 3.1. *If there is a unique min weight perfect matching of G (call it M) then:*

- $\det(A) \neq 0$
- *The highest power of 2 that divides* $\det(A)$ *is* 2^{2W}, *where W is the weight of M. I.e.* $\det(A) = 2^{2W} \times$ *(an odd number).*

Proof. Look at the contributions of various permutations π to $\det(A)$:

- If π has an odd cycle then, just as argued before (by reversing the direction of the "first" odd cycle), its contribution to the determinant is cancelled out.

- 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 $W_1 + W_2 > 2W$. ■

Assuming M is unique we proceed as follows to find M (if M is not unique this may not work but we will be able to detect the failure and repeat): let

$$\begin{aligned} m_{ij} &= \sum_{\pi:\pi(i)=j} (-1)^{\pi} \prod_{k=1}^{n} A_{k,\pi(k)} \\ &= \pm 2^{w_{ij}} \det(\hat{A}_{ij}) \end{aligned} \tag{7}$$

where $\hat{A}_{ij}$ is the determinant of the (i,j) minor of A (the matrix obtained by removing the i'th row and j'th column from A).

Claim 3.2. *For every $(i,j) \in E$:*

1. *The total contribution to m_{ij} of permutations π having odd cycles is 0.*
2. *If $(i,j) \in M$ then $m_{ij}/2^{2W}$ is odd.*
3. *If $(i,j) \notin M$ then $m_{ij}/2^{2W}$ is even.*

Proof.

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 an odd number of signs have been flipped (in terms of the entries we are using from the Tutte matrix).
2. By (1), we need only consider permutations containing solely even cycles. There is exactly one such contribution of value $\pm 2^{2W}$. Just as argued for 3.1, all other permutations contribute $\pm 2^{C}$, for values $C > 2W$.
3. Again by (1) we need only consider permutations containing solely even cycles, and again as in claim 3.1, each of these contributes $\pm 2^{C}$ for various $C > 2W$. ■

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

1. Generate the weights w_i uniformly in $\{1, \ldots, n^2\}$.
2. Define A as in equation (6), compute its determinant and if it is nonsingular invert it.

3. Determine W by factoring the greatest power of 2 out of $\det(A)$.
4. Obtain the values $\pm m_{ij}$ from the equations $m_{ij} = \pm 2^{w_{ij}} \det(\hat{A}_{ij})$ and $\det(\hat{A}_{ij}) = (-1)^{i+j}(A^{-1})_{ji} \det(A)$. 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, generate new weights and repeat the process.

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

Since the probability that the minimum weight perfect matching (if any exist) is not unique is at most 1/2, the expected number of repetitions of the procedure is at most 2. (If a 0 determinant is encountered then it may be that there are no perfect matchings; one may first run the Rabin-Vazirani procedure in order to get a high likelihood of weeding out that case.)

The numbers in the matrix A are integers bounded by $\pm 2^{n^2}$. The determinant and inversion procedures (carried out in rational arithmetic) can involve integers up to size about 2^{n^3}. Rational arithmetic can be executed in nearly linear time in the number of bits required to represent the integers. Hence the time for a single arithmetic operation in this matrix is bounded by approximately n^3. Many matrix computations, including multiplication, inversion and determinant computation, have equivalent asymptotic complexity, bounded by n^ω, where ω is unknown; the best known upper bound is about 2.376, but no lower bound better than 2 is known. The overall runtime of the MVV algorithm is therefore $O(n^{\omega+3})$ (up to polylogarithmic terms).

In fact if we examine the use of the isolating lemma we see that the weights should really be chosen between 1 and $2|E|$ where E is the edge set of G, so in fact the algorithm modified in this way gives:

Theorem 1. *The MVV algorithm finds a perfect matching, if one exists, in expected time $O(n^{\omega+1}|E|)$.*

Matrix inversion is fairly efficiently parallelizable: it can be performed in time $O(\log^2 n)$ on $O(n^\omega)$ processors. (Incidentally this is not quite as good as matrix multiplication, which can be performed in time $O(\log n)$ on $O(n^\omega)$ processors.) This, together with the fact that integer arithmetic can be efficiently parallelized, implies that the above method has a work-preserving (up to log factors) RNC implementation. (For detail see [2] §2.4 & 2.5.5.)

References

[1] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers.* Oxford, fifth edition, 1979.

[2] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes.* Morgan Kaufmann, 1992.

[3] K. Mulmuley, U. V. Vazirani, and V. V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7:105–113, 1987.

[4] M. O. Rabin and V. V. Vazirani. Maximum matchings in general graphs through randomization. *J. Algorithms*, 10(4):557–567, 1989.