Chapter 5

Randomized Algebraic Algorithms

Chapter by J. Matoušek (with small additions by E. Welzl).

Keywords: matrix multiplication, Monte Carlo algorithm, probabilistic checking, zero-testing of polynomial, Schwartz-Zippel Theorem, perfect matching, permanent, determinant, finite field, Tutte matrix, cycles of a permutation.

In this chapter we consider algorithms based on the idea of *probabilistic checking*, which proved extremely fruitful in computer science. It led to the celebrated *PCP theorem*, which essentially says that correctness of every solution of a problem in NP can be checked extremely fast (well, if the solution is written in the appropriate way).

We will first illustrate probabilistic checking on a very simple example, and then, after some preparations, we will consider randomized algorithms for testing the existence of perfect matchings in a graph.

5.1 Checking Matrix Multiplication

Multiplying two $n \times n$ matrices is a very important operation. Moreover, many other matrix operations, such as inversion or determinant, can be computed in asymptotically the same time as matrix multiplication.

A straightforward algorithm multiplying two $n \times n$ matrices requires about n^3 arithmetic operations, but ingenious algorithms have been discovered that can do such a multiplication in an asymptotically much better time. The current record is an $O(n^{2.376})$ algorithm. The constant of proportionality is so astronomically large that the algorithm is only the-

oretically interesting, since matrices for which it would prevail over the straightforward algorithm do not fit into any existing or future computer.

But progress cannot be stopped and soon a software company may start selling a program called MATRIX WIZARD that, supposedly, multiplies matrices real fast. Since wrong results could be disastrous for you, you would like to have a simple *checking program* appended to MATRIX WIZARD that would always check whether the resulting matrix C is really the product of the input matrices A and B. Of course, a checking program that would actually multiply A and B and compare the result with C makes little sense, since you do not know how to multiply matrices as fast as MATRIX WIZARD. But it turns out that if we allow for some slight probability of error in the checking, there is a very simple and efficient checker for matrix multiplication.

Let us first consider the ad-hoc method, just to appreciate even more what comes next. The checking algorithm receives $n \times n$ matrices A, B, C as the input. In order to check $C = AB$ we can simply choose ℓ entries c_{ij} of C (u.a.r. from all ℓ-element subsets of all n^2 entries) and verify that indeed

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

holds. That takes time $\Theta(n)$ for each entry, i.e. $\Theta(\ell n)$ time altogether. If C is wrong in just one entry, then the probability of our ℓ random choices capturing this entry is[1]

$$\frac{\binom{n^2-1}{\ell-1}}{\binom{n^2}{\ell}} = \frac{\ell}{n^2} .$$

That is, even if we choose $\ell = n$, thereby spending time $\Theta(n^2)$, we are still left with a small success probability of $\frac{1}{n}$. After all, if a single entry is wrong, it's like looking for a needle in a haystack, so what can we expect. Much more, as we will see.

For a next attempt, we assume that the considered matrices consist of rational numbers, although everything works without change for matrices

[1] An even more direct way of seeing this probability $\frac{\ell}{n^2}$ is as follows. For reasons of symmetry, we may as well fix the ℓ entries to check and take the faulty entry u.a.r. from the n^2 entries—so the probability for this entry to fall among the ℓ entries checked is clearly $\frac{\ell}{n^2}$.

over any field. Using a random number generator, it picks a random n-component vector x of zeros and ones. More precisely, each vector in $\{0,1\}^n$ appears with the same probability, equal to 2^{-n}. The algorithm computes the products Cx (using $O(n^2)$ operations) and ABx (again with $O(n^2)$ operations; the right parenthesizing is, of course, $A(Bx)$). If the results agree, the algorithm answers YES, otherwise it answers NO.

If $C = AB$, the algorithm always answers YES, which is correct. But if $C \neq AB$, it can answer both YES and NO. We claim that a wrong answer YES has probability at most $\frac{1}{2}$, and thus the algorithm detects a wrong matrix multiplication with probability at least $\frac{1}{2}$.

Let us set $D := C - AB$. It suffices to show that if D is any nonzero $n \times n$ matrix and $x \in \{0,1\}^n$ is random, then the vector $y := Dx$ is zero with probability at most $\frac{1}{2}$.

Suppose that $d_{ij} \neq 0$. We want to derive that then the probability of $y_i = 0$ is at most $\frac{1}{2}$. We have

$$y_i = d_{i1}x_1 + d_{i2}x_2 + \cdots + d_{in}x_n = d_{ij}x_j + S,$$

where

$$S := \sum_{\substack{k=1,2,\ldots,n \\ k \neq j}} d_{ik}x_k.$$

Imagine that we choose the values of the entries of x according to successive coin tosses and that the toss deciding the value of x_j is made as the last one (since the tosses are independent it doesn't matter). Before this last toss, the quantity S is already fixed, because it doesn't depend on x_j. After the last toss, we either leave S unchanged (if $x_j = 0$) or add the nonzero number d_{ij} to it (if $x_j = 1$). In at least one of these two cases, we must obtain a nonzero number, and so $y_i = 0$ has probability at most $\frac{1}{2}$ as claimed.

The described checking algorithm, although a definite progress compared to the previous, is still not very reliable: It may fail to detect an error with probability as high as $\frac{1}{2}$. But if we repeat it, say, fifty times for a single input A, B, C, it fails to detect an error with probability at most $2^{-50} < 10^{-15}$, and this probability is totally negligible for practical purposes.

Exercise 5.1. Checking over GF(2)

Suppose we are running the checking algorithm for matrices over GF(2), *i.e. numbers are* $\{0,1\}$ *with addition and multiplication* mod 2. *Show*

that in one iteration the success probability of detecting an error in the supposed product matrix C *is exactly* $\frac{1}{2}$*, in case matrix* C *is wrong in exactly one row.*

Exercise 5.2. For the Letter of Complaint
If the matrix multiplication checking algorithm detects the existence of an error, how much extra time does it take to explicitely find an entry in the supposed product that is wrong (after all, we have to point at the problem when complaining to the MATRIX WIZARD company).

Exercise 5.3. More Vectors—Smaller Error
Consider a modified checker for matrix multiplication (over $\mathbf{Q}$*): Instead of choosing a random vector with components* 0 *and* 1*, choose a random vector with components drawn from* $\{0, 1, \ldots, N-1\}$ *uniformly and independently at random. Show that the probability of* failure *(declaring an incorrect multiplication correct) is at most* $\frac{1}{N}$.

REMARK: Of course, we still pay a prize. While multiplication with 0 or 1 is for free, that does take some time as the multipliers get larger.

5.2 Is a Polynomial Identically Zero?

How can one tell whether a given polynomial is identically zero? This is easy if the polynomial is given to us by the coefficients.[2] But instead of the coefficients we may be given a black box that evaluates the polynomial at any given point. This is not a completely unusual situation, as we will see later.

For an arbitrary function given by a black box we can never be sure that it is zero everywhere unless we evaluate it at all points, but polynomials have a remarkable "rigidity" property: A low-degree polynomial either is zero everywhere, or it is nonzero almost everywhere. Slightly more generally but equivalently, two low-degree polynomials either coincide or they differ almost everywhere.

Let us first talk about polynomials in a single variable (so-called *univariate polynomials*), although this is not quite the problem we mean. A

[2] Well, complications may arise for finite fields, since for example, the polynomial x^2+x is identically 0 over the two-element field GF(2). But at least for infinite fields things are clear, since a polynomial is identically zero if and only if all coefficients are 0.

simple but fundamental result from algebra tells us that if a nonzero univariate polynomial with coefficients from a field has degree at most d, then it has at most d zeros. Therefore, it suffices to have such a polynomial evaluated at any $d+1$ points: If it is 0 at all of them, then it must be identically 0, and otherwise, of course, it is nonzero.

The situation for polynomials in more than one variable is not so simple, since the zero set of a multivariate polynomial can be rather complicated (for example, the zero set of the polynomial $x_1^2+x_2^2-1$ is the unit circle, the zero set of x_1x_2 is the union of two lines, etc.). If we want to be absolutely sure that a polynomial in many variables is zero, we have to evaluate it in quite many points. But using randomization, we can be almost sure after evaluating it only a few times.

Let us now consider a polynomial in n variables $x_1, x_2, \ldots, x_n$, such as $3x_1^6x_2^2x_5 - x_3^2x_4^5 + 2x_5^2 - 17$ (this one is in 5 variables, but if desired, we could also regard it as a polynomial in 27 variables, where x_6 through x_{27} don't show up). A general polynomial in n variables is a finite sum of terms of the form $a_{i_1,i_2,\ldots,i_n}x_1^{i_1}x_2^{i_2}\cdots x_n^{i_n}$ (called *monomials*), where the exponents $i_1, \ldots, i_n$ are nonnegative integers and $a_{i_1,i_2,\ldots,i_n}$ is a coefficient. The *degree* of this term is $i_1+i_2+\cdots+i_n$, and the *degree of a polynomial* is the maximum of the degrees of its terms (with nonzero coefficients). For example, the polynomial above has degree 9 because of the first term.

The reader will probably be most familiar with polynomials with rational or real coefficients. Here we will allow for polynomials with coefficients from an arbitrary field $\mathbb{F}$. This is actually important in algorithmic applications; it turns out that by working with suitable finite fields one can sometimes do significantly better than by working with rational coefficients. Let $\mathbb{F}[x_1, x_2, \ldots, x_n]$ denote the ring of all polynomials in the variables $x_1, x_2, \ldots, x_n$ with coefficients in $\mathbb{F}$.

Theorem 5.1 (Schwartz–Zippel theorem).[3]
Let $p(x_1, \ldots, x_n) \in \mathbb{F}[x_1, \ldots, x_n]$ be a (nonzero) polynomial of degree $d \geq 0$, and let $S \subseteq \mathbb{F}$ be a finite set. Then the number of n-tuples $(r_1, r_2, \ldots, r_n) \in S^n$ with $p(r_1, r_2, \ldots, r_n) = 0$ is at most $d|S|^{n-1}$. In other words, if $r_1, \ldots, r_n \in S$ are chosen independently and uniformly at random, then the probability of $p(r_1, r_2, \ldots, r_n) = 0$ is at most $\frac{d}{|S|}$.

Proof. We proceed by induction on n. The univariate case is clear, since

[3]This Schwartz is really with "t", unlike the one from the Cauchy–Schwarz inequality.

there are at most d roots of $p(x_1)$.

Let $n > 1$. Let us suppose that x_1 occurs in at least one term of $p(x_1, \ldots, x_n)$ with nonzero coefficient (if not, we rename the variables). Let us write $p(x_1, \ldots, x_n)$ as a polynomial in x_1 with coefficients in $\mathbb{F}[x_2, \ldots, x_n]$:

$$p(x_1, x_2, \ldots, x_n) = \sum_{i=0}^{k} x_1^i \, p_i(x_2, \ldots, x_n),$$

where k is the maximum exponent of x_1 in $p(x_1, \ldots, x_n)$.

We classify the n-tuples $(r_1, r_2, \ldots, r_n)$ with $p(r_1, \ldots, r_n) = 0$ into two groups. The first group are those with $p_k(r_2, \ldots, r_n) = 0$. Since the polynomial $p_k(x_2, \ldots, x_n)$ is not identically zero and has degree at most $d-k$, the number of choices for $(r_2, \ldots, r_n)$ is at most $(d-k)|S|^{n-2}$ by the induction hypothesis, and so the first group has at most $(d-k)|S|^{n-1}$ n-tuples.

The second group are the remaining n-tuples, with $p(r_1, r_2, \ldots, r_n) = 0$ but $p_k(r_2, \ldots, r_n) \neq 0$. Here we count as follows: r_2 through r_n can be chosen in at most $|S|^{n-1}$ ways, and if $r_2, \ldots, r_n$ are fixed with $p_k(r_2, \ldots, r_n) \neq 0$, then r_1 must be a root of the univariate polynomial $q(x_1) := p(x_1, r_2, \ldots, r_n)$. This is a nonzero polynomial of degree k, and hence it has at most k roots. The second group therefore has at most $k|S|^{n-1}$ n-tuples, which gives $d|S|^{n-1}$ altogether, finishing the induction step. □

Exercise 5.4. Proving Schwartz-Zippel Tight

Given a finite set S of rational numbers and positive integers d and n, $d \leq |S|$, find a polynomial $p(x_1, x_2, \ldots, x_n)$ of degree d for which the Schwartz–Zippel theorem is tight. That is, the number of n-tuples $(r_1, \ldots, r_n) \in S^n$ with $p(r_1, \ldots, r_n) = 0$ is $d|S|^{n-1}$.

Exercise 5.5. Extracting Polynomials

Suppose that a polynomial $p(x_1, \ldots, x_n, y)$ in $n+1$ variables is given by a black box that, given concrete values for $x_1, \ldots, x_n$ and y, returns the value of the polynomial. Let m be the maximum degree of y in p and write $p(x_1, \ldots, x_n, y) = \sum_{i=0}^{m} y^i p_i(x_1, \ldots, x_n)$. Given an integer k and numbers $r_1, \ldots, r_n$, how can we compute $p_k(r_1, \ldots, r_n)$ (using only the black box)? Assume we know the degrees of the polynomials $p_i(x_1, \ldots, x_n)$ or at least some upper bounds. How can we test whether $p_k(x_1, \ldots, x_n)$ is a nonzero polynomial? For simplicity, assume that everything happens over the rationals.

5.3 Testing for Perfect Bipartite Matchings

We want to know whether a graph G has a *perfect matching*, i.e. a matching covering all vertices. A maximum matching can be computed in time $O(m\sqrt{n})$, but the known algorithms are rather complicated. Here we explain very simple randomized algorithms for testing the existence of a perfect matching. These algorithms use the Schwartz–Zippel theorem for testing whether a suitable polynomial associated to the considered graph is zero, and the polynomial is given as the determinant of a certain matrix.

We begin with the bipartite case. Let us consider a bipartite graph G with color classes $\{u_1, u_2, \ldots, u_n\}$ and $\{v_1, v_2, \ldots, v_n\}$. Let $\mathcal{S}_n$ denote the set of all permutations of $\{1, 2, \ldots, n\}$. We note that a perfect matching in G, if one exists, corresponds to a permutation $\pi \in \mathcal{S}_n$: It has the form $\{\{u_1, v_{\pi(1)}\}, \{u_2, v_{\pi(2)}\}, \ldots, \{u_n, v_{\pi(n)}\}\}$. We introduce an $n \times n$ matrix B by letting

$$b_{ij} := \begin{cases} 1 & \text{if } \{u_i, v_j\} \in E(G) \\ 0 & \text{otherwise.} \end{cases}$$

A given permutation π determines a perfect matching in G if and only if the product $b_{1,\pi(1)} b_{2,\pi(2)} \cdots b_{n,\pi(n)}$ equals 1, and so the sum

$$\operatorname{per}(B) := \sum_{\pi \in \mathcal{S}_n} b_{1,\pi(1)} b_{2,\pi(2)} \cdots b_{n,\pi(n)},$$

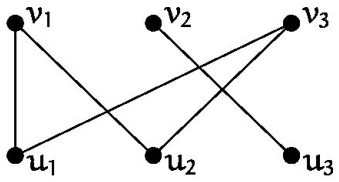
called the *permanent* of B, counts all perfect matchings of G. Our problem is thus to test whether $\operatorname{per}(B) = 0$.

The definition of $\operatorname{per}(B)$ resembles the definition of $\det(B)$ but the determinant has the sign of π in front of each product. This innocent-looking difference is crucial: While the determinant can be computed in polynomial time, the computation of the permanent is NP-hard (even #P-complete, for those more advanced in complexity theory). We can have $\det(B) = 0$ even if $\operatorname{per}(B) \neq 0$, since nonzero terms may cancel out in the determinant.

To avoid such cancellations, we introduce one variable (indeterminate) x_{ij} for each edge $\{u_i, v_j\} \in E(G)$, and we define another matrix A:

$$a_{ij} := \begin{cases} x_{ij} & \text{if } \{u_i, v_j\} \in E(G) \\ 0 & \text{otherwise.} \end{cases}$$

So A is a function of the x_{ij}, and $\det(A)$ is a polynomial in these $|E(G)|$ variables. Since the determinant is a sum of products of n variables each, the degree of $\det(A)$ is n (unless the polynomial is constant 0).



$$A = \begin{pmatrix} x_{11} & 0 & x_{13} \\ x_{21} & 0 & x_{23} \\ 0 & x_{32} & 0 \end{pmatrix}$$

Figure 5.1: A graph and its matrix A with $\det(A) = -x_{11}x_{23}x_{32}+x_{13}x_{21}x_{32}$— the two monomials of the determinant correspond to the two perfect matchings in the graph.

Lemma 5.2. *The graph G has a perfect matching if and only if the polynomial $\det(A)$ is not identically zero.*

Proof. Each monomial of $\det(A)$ with nonzero coefficient determines a perfect matching in G. Conversely, if G has a perfect matching M, we substitute 1 for x_{ij} if the edge $\{u_i, v_j\}$ is in M and 0 otherwise, and we get a matrix with determinant ± 1. □

The determinant of A can thus be used for testing whether G has a perfect matching. We cannot afford to compute it (as a polynomial), since it can have exponentially many terms, but we can use the Schwartz–Zippel theorem.

Since $\deg(\det(A)) \leq n$, we need a set S of size larger than n, say at least $2n$, for choosing the random values of the x_{ij}. (Recall here that Schwartz–Zippel tells us that the error probability is at most $\frac{\deg(\det(A))}{|S|}$; so this better be small.) The first attempt might be to consider $\det(A)$ as a polynomial with rational coefficients and to use $S = \{1, 2, \ldots, 2n\}$. But to decide whether the determinant is 0 for a given substitution, we have to compute exactly. We might encounter numbers with about n bits in the computation, and so the arithmetic operations would become quite expensive.

It is better to work with a finite field. The simplest way is to choose a prime number p, $2n \leq p < 4n$, and to regard $\det(A)$ as a polynomial with coefficients in $GF(p)$, the field of integers modulo p. Then

- It is still true that $\det(A)$ is nonzero if and only if G has a perfect matching; the proof of Lemma 5.2 works over any field.
- We can choose the random values for the x_{ij} among at least $2n$ elements, and so the probability of detecting that $\det(A)$ is nonzero is

at least $\frac{1}{2}$.

- The determinant of a matrix over GF(p) can be computed using $O(n^3)$ arithmetic operations by Gaussian elimination, or even using $O(M(n)) = O(n^{2.376})$ arithmetic operations, where $M(n)$ is the number of operations required to multiply two $n \times n$ matrices. The arithmetic operations in GF(p) are fast if we prepare a table of inverse elements in advance.

- Finally, a prime p between $2n$ and $4n$ always exists by a theorem from number theory called Bertrand's postulate (even stronger results are known). It can be found in $O(n^{3/2})$ time by testing all numbers one by one, by the simplest primality test anyone can come up with (or also much faster by more advanced methods, but we don't need that here).

Summarizing, we have an algorithm that can test whether a given bipartite graph has a perfect matching, has running time $O(M(n))$, and has failure probability at most $\frac{1}{2}$. As usual, the failure probability can be made smaller than any given $\delta > 0$ by $O(\log \frac{1}{\delta})$ repetitions.

Exercise 5.6. Perfect Matchings Parity

Describe an efficient deterministic algorithm for deciding whether a bipartite graph has an odd number of perfect matchings.

HINT: Consider B as a matrix over GF(2).

5.4 Perfect Matchings in General Graphs

Permutations, Cycles, Signs. We briefly recapitulate some basic notions about permutations in order to have them handy when we will need them later. S_n is the set of all bijective mappings $\{1..n\} \to \{1..n\}$. This set equipped with composition forms a group generated by transpositions (i, j), defined as

$$i \mapsto j, j \mapsto i, \quad \text{and, for } k \notin \{i, j\}, \quad k \mapsto k\,.$$

Although a permutation can be written in several ways as composition of transpositions, the parity of the number of transpositions in a representation of a permutation π is an invariant of π and it defines the sign of π: $\text{sign}(\pi) = +1$ if the number of transpositions is even, and $\text{sign}(\pi) = -1$, otherwise.

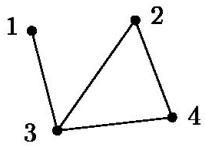
Given a permutation $\pi \in \mathcal{S}_n$ the set $\{(i, \pi(i)) | i \in \{1..n\}\}$ constitutes a set of directed edges on the vertex set $\{1..n\}$. Clearly, every vertex has out- and in-degree 1, and therefore it partitions $\{1..n\}$ into cycles (some may be cycles of length 1: loops; e.g. for the identity map we get n loops). It turns out that the sign of a permutation is

$$(-1)^{\text{number of even cycles}},$$

i.e. the sign is positive iff the number of even cycles is even. (A cycle $i_1 \mapsto i_2 \mapsto \ldots \mapsto i_k \mapsto i_1$ can be written as composition $(i_1, i_k) \circ (i_1, i_{k-1}) \cdots \circ (i_1, i_2)$; therefore, if k is even, this gives an odd number of transpositons.)

The algorithm from the previous section can be extended to arbitrary, not necessarily bipartite, graphs, but we need to work with another matrix. For a graph G with a vertex set $V = \{v_1, v_2, \ldots, v_n\}$ we introduce a variable x_{ij} for every $1 \leq i < j \leq n$ with $\{v_i, v_j\} \in E(G)$, and we define the *Tutte matrix* A of G by

$$a_{ij} := \begin{cases} +x_{ij} & \text{if } i < j \text{ and } \{v_i, v_j\} \in E(G), \\ -x_{ji} & \text{if } i > j \text{ and } \{v_i, v_j\} \in E(G), \\ 0 & \text{otherwise.} \end{cases}$$



$$\begin{pmatrix} 0 & 0 & x_{13} & 0 \\ 0 & 0 & x_{23} & x_{24} \\ -x_{13} & -x_{23} & 0 & x_{34} \\ 0 & -x_{24} & -x_{34} & 0 \end{pmatrix}$$

Figure 5.2: A graph and its Tutte matrix.

Theorem 5.3 (Tutte). *The polynomial* $\det(A)$ *is nonzero if and only if* G *has a perfect matching.*

Proof. One direction is equally easy as in the bipartite case (Lemma 5.2): If G has a perfect matching M, then we substitute $x_{ij} = 1$ if $\{v_i, v_j\}$ is in the matching and $x_{ij} = 0$ otherwise. Since every vertex v_i is connected in M to exactly one other vertex, the resulting matrix has exactly one nonzero entry, $+1$ or -1, in every row and in every column, and so the determinant is ± 1.

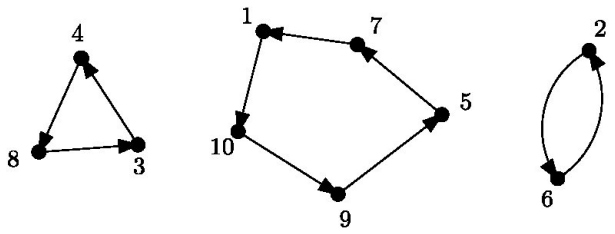
The opposite implication is considerably harder than for the bipartite case, because it is no longer true that nonzero terms in the expansion of the

$\det(A)$ are contributed only by perfect matchings. As a simple example, let us consider the triangle K_3, whose Tutte matrix is

$$A = \begin{pmatrix} 0 & x_{12} & x_{13} \\ -x_{12} & 0 & x_{23} \\ -x_{13} & -x_{23} & 0 \end{pmatrix}.$$

Expanding $\det(A)$ according to the definition of determinant gives $\det(A) = +x_{12}x_{23}(-x_{13}) + x_{13}(-x_{12})(-x_{23})$. This yields 0 as it should, since the two terms "miraculously" cancel out, but we must show that all terms that do not come from a perfect matching *always* cancel out.

To this end, we need to talk about cycles of a permutation $\pi \in S_n$: We define the *graph* of π as the directed graph with vertex set $\{1, 2, \ldots, n\}$ and with edge set $\{(i, \pi(i)) : i = 1, 2, \ldots, n\}$; for example:



Every vertex has indegree 1 and outdegree 1, and so this graph consists of disjoint cycles, possibly also of lengths 1 (loops) and 2 (two opposite edges).

Let us see what terms contribute to the determinant of the Tutte matrix A. By definition

$$\det(A) = \sum_{\pi \in S_n} \operatorname{sign}(\pi) \cdot a_{1,\pi(1)} a_{2,\pi(2)} \cdots a_{n,\pi(n)}.$$

Let us write $a(\pi) = a_{1,\pi(1)} a_{2,\pi(2)} \cdots a_{n,\pi(n)}$, and let us call π *important* if $a(\pi) \neq 0$. Let us define, for a permutation $\pi \in S_n$,

$$E_\pi := \Big\{\{v_i, v_{\pi(i)}\} : i = 1, 2, \ldots, n\Big\}.$$

So E_π is an undirected version of the graph of π introduced above, and π is important if and only if $E_\pi \subseteq E(G)$. In particular, we note that the graph of an important π has no loops (cycles of length 1).

We want to show that *if* $\det(A) \neq 0$, *then there is at least one important* π *with all cycles of even length.* This is sufficient since if all cycles

have even length, then E_π contains a perfect matching, and thus G has a perfect matching as well.

Let OI denote the set of all important permutations (for G) that have at least one odd cycle. We define a mapping $r : OI \to OI$ as follows: Among all odd cycles of a permutation $\pi \in OI$, we consider the one containing the smallest number, and we form the permutation $r(\pi)$ by reversing arrows on that cycle. For example, for the permutation π in the above picture, the corresponding $r(\pi)$ is given by $1 \mapsto 7$, $2 \mapsto 6$, $3 \mapsto 4$, $4 \mapsto 8$, $5 \mapsto 9$, $6 \mapsto 2$, $7 \mapsto 5$, $8 \mapsto 3$, $9 \mapsto 10$, and $10 \mapsto 1$. Clearly, r is a bijection, and it is easy to check that $a(r(\pi)) = -a(\pi)$ (here the signs in the Tutte matrix come into play). With a little more work one can show that $\operatorname{sign}(r(\pi)) = \operatorname{sign}(\pi)$ for all $\pi \in OI$. So $\sum_{\pi \in OI} \operatorname{sign}(\pi) \cdot a(\pi) = 0$, and hence if $\det(A) \neq 0$, there has to be an important permutation with all cycles of even length as claimed. □

Using Tutte's theorem and the tools from the previous sections, we obtain a randomized algorithm for testing whether a given graph has a perfect matching, with $O(M(n))$ running time and probability of failure at most $\frac{1}{2}$.

Exercise 5.7. Signs of Permutations and Odd Cycles
Let $n \in \mathbf{N}$. The sign $\operatorname{sign}(\pi)$ of a permutation π of $\{1..n\}$ can be defined, e.g., by

$$\operatorname{sign}(\pi) = \prod_{1 \le i < j \le n} \frac{\pi(i) - \pi(j)}{i - j}.$$

Recall that for two permutations π and σ of $\{1..n\}$ we have

$$\operatorname{sign}(\pi \circ \sigma) = \operatorname{sign}(\pi) \cdot \operatorname{sign}(\sigma),$$

where $\circ$ is the concatenation of π and σ, i.e $\pi \circ \sigma(x) = \pi(\sigma(x))$.

Furthermore recall that for a permutation π with an odd cycle the permutation $r(\pi)$ has exactly the smallest cycle reversed.

(a) Show that for a permutation π which consists of only one odd cycle and no even cycle the signs of π and $r(\pi)$ are equal.

(b) Show that for a permutation π with an odd cycle the signs of π and $r(\pi)$ agree.

Exercise 5.8. The Permanent and the Determinant
Let A *be an* $n \times n$ *matrix with* 0/1*-entries. For* $1 \leq i, j \leq n$ *let* $\epsilon_{i,j}$ *be independent random variables,* $\epsilon_{i,j} \in_{u.a.r.} \{-1, +1\}$. *Let* B *be the random matrix with* $b_{i,j} = \epsilon_{i,j} \cdot a_{i,j}$. *In other words, to get* B *from* A *we randomly assign signs to the entries of* A.

(a) Show that $\mathbf{E}[\det B] = 0$.

(b) Show that $\mathbf{E}\left[(\det B)^2\right] = \mathrm{per}(A)$.

REMARK: (b) may be challenging.

5.5 Comparison with Other Matching Algorithms

As we have remarked, the best known deterministic algorithm computes a maximum matching in an arbitrary graph in $O(m\sqrt{n})$ time, where n is the number of vertices and m is the number of edges.

Running Time. With the theoretically fastest known matrix multiplication, the described randomized algorithm can *test* the existence of a perfect matching in time roughly $O(n^{2.376})$. This is the asymptotically fastest known method, even for the bipartite case. With Gaussian elimination the algorithm works reasonably well in practice and it is very easy to code. On the other hand, the deterministic algorithm is theoretically faster for not too dense graphs.

Testing versus Finding. If a perfect matching exists, we usually also want to find one. The deterministic algorithm does it, while for the randomized one computing a perfect matching requires additional work. There is a simple way of using any testing algorithm for computing a perfect matching (the reader is invited to try this as Exercise 5.9), but efficiency is lost since the testing algorithm needs to be called many times.

A more clever and more complicated extension of the testing algorithm, discovered in 2004, can *find* a perfect matching in a general graph in $O(M(n))$ time if one exists, where $M(n)$ is the time needed for multiplying two $n \times n$ matrices (with polynomially large integer entries). For dense graphs, this is again the theoretically fastest known algorithm.

Maximum Matching. The deterministic algorithm can also find a maximum matching (which is not necessarily perfect). The randomized algorithm can also be extended to do this. For example, to find a matching of size k in a graph G, we can form a new graph G^* by adding $n - 2k$ new vertices and connecting them to all old vertices. It is easy to see that G^* has a perfect matching if and only if G has a matching with k edges, and any perfect matching in G^* yields a matching with k edges in G. To find a maximum matching, we can do binary search over k.

The sophisticated $O(M(n))$ perfect matching algorithm mentioned above also works for maximum matchings.

Parallel Algorithms. The randomized approach via determinants can be efficiently parallelized; namely, it yields a parallel algorithm for testing the existence of a perfect matching that runs on polynomially many processors in $O((\log n)^{\text{const}})$ time. (Problems admitting such a parallel algorithm are said to belong to the class NC or, if the algorithm is randomized as in our case, RNC.) This requires a fast parallel computation of the determinant, which is nontrivial but possible. No fast parallel versions are known for deterministic maximum matching algorithms.

Colored Matching. Finally, the randomized approach discussed above can also solve other matching-type problems for which no polynomial-time deterministic algorithms are known. For example, suppose that each edge of a given bipartite graph is colored red or blue. Does there exist a perfect matching using exactly k red edges? The reader may try to extend the algorithm from Section 5.3 to this problem (see Exercise 5.10).

Exercise 5.9. Existence vs. Explicit Construction
Suppose that we have an algorithm for testing the existence of a perfect matching in a given graph, with running time at most $T(n)$ for any n-vertex graph.

(a) Explain how repeated calls to the algorithm can be used to find a perfect matching if one exists. Estimate the running time of the resulting algorithm.

(b) How can the algorithm be used for finding a maximum matching in a given graph?

Exercise 5.10. Just Enough Red

Consider a bipartite graph in which some edges are colored red and some blue. Extend the randomized algorithm discussed in class to handle the following problem: Given such a colored bipartite graph and an integer k, is there a perfect matching that contains exactly k red edges?

HINT: Use Exercise 5.5.

REMARK: No polynomial-time deterministic algorithm for this problem seems to be known.

5.6 Counting Perfect Matchings in Planar Graphs[4]

Sometimes it is even possible to compute the number, $\mathrm{pm}(G)$, of perfect matchings in a graph G by looking at the determinant of the "right" matrix derived from the graph.

Let $G = (V_G, E_G)$, $V_G = \{1..n\}$, be a graph with n even and let $\vec{G} = (V_G, E_{\vec{G}})$ be an orientation of G, i.e. every edge $\{i,j\}$ in G is represented by exactly one of (i,j) or (j,i) in $\vec{G}$. For such an orientation we define the following matrix

$$A_s(\vec{G}) = (a_{ij})_{i,j=1}^{n} \in \{0,+1,-1\}^{n\times n}\text{, where}$$

$$a_{ij} := \begin{cases} +1 & \text{if } (i,j) \in E_{\vec{G}}, \\ -1 & \text{if } (j,i) \in E_{\vec{G}}, \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

This is what is called a skew-symmetric matrix (meaning that $a_{ij} = -a_{ji}$ for all (i,j), in particular $a_{ii} = 0$). Note that the matrix is nothing else but the previously considered Tutte matrix A of G where we have plugged in $+1$ or -1 for the variables x_{ij}, $i < j$, $\{i,j\} \in E_G$. So some of what we have learned for the Tutte matrix will be useful here.

A Lower Bound for the Number of Perfect Matchings.

Lemma 5.4.

$$\det(A_s(\vec{G})) \leq \mathrm{pm}(G)^2$$

[4]Addendum by E. Welzl (following L. Lovász, M.D. Plummer, *Matching Theory*, in: Annals of Discrete Mathematics Vol. 29, North-Holland,1986).

Proof. For perfect matchings M_1 and M_2 in G, consider their union $U := M_1 \cup M_2$. This is an edge set $U \subseteq E_G$ consisting of even cycles and independent edges. (If $M_1 = M_2$, then U is a perfect matching itself.) Given such a $U = M_1 \cup M_2$, for how many pairs (M', M'') of perfect matchings in G can we write $U = M' \cup M''$? This is exactly 2^k, where k is the number of cycles in U (not counting the independent edges), because the edges of an even cycle partition into two perfect matchings of its vertex set and we can choose for M' one of the two for each even cycle in U. So if $\mathcal{U}$ denotes the set of all unions of two perfect matchings in G and $||U||$, for $U \in \mathcal{U}$, is the number of even cycles in U, then

$$\mathrm{pm}(G)^2 = \sum_{U \in \mathcal{U}} 2^{||U||} .$$

Note that for a permutation $\pi \in \mathcal{S}_n$ with all of its cycles even, the previously defined set $E_\pi = \{\{i, \pi(i)\} \mid i \in \{1..n\}\}$ is of the form that it decomposes into even cycles and independent edges (for the cycles of length 2 in π). It easily follows that

$$E_\pi \subseteq E_G \quad \Leftrightarrow \quad E_\pi \in \mathcal{U} .$$

Given $U \in \mathcal{U}$, for how many permutations π do we have $E_\pi = U$? Interestingly, this is again $2^{||U||}$ since we have for each cycle of length at least 4 exactly two possibilities to orient the cycle – for the independent edges in U there is no choice for the cycle of length two associated with it. So we can extend the equivalence above to

$$\mathrm{pm}(G)^2 = \sum_{U \in \mathcal{U}} 2^{||U||} = |\mathrm{IE}| .$$

where $\mathrm{IE} := \{\pi \in \mathcal{S}_n \mid E_\pi \subseteq E_G, \text{all cycles in } \pi \text{ even}\}$, the set of important permutations with all cycles even. Now we inspect the determinant. We have argued before that $a(\pi) := a_{1\pi(1)} a_{2\pi(2)} \cdots a_{n\pi(n)} \neq 0$ (i.e. $+1$ or -1) iff $E_\pi \subseteq G$ and that the terms of such permutations with odd cycles cancel each other out (This was true in the Tutte matrix with variables, so it is also true, of course, if we plug in some values for the variables). Therefore

$$\begin{aligned} \det(A_s(\vec{G})) &= \sum_{\pi \in \mathcal{S}_n} \mathrm{sign}(\pi) a_{1\pi(1)} a_{2\pi(2)} \cdots a_{n\pi(n)} \\ &= \sum_{\pi \in \mathrm{IE}} \mathrm{sign}(\pi) \underbrace{a(\pi)}_{=\pm 1} \\ &\leq |\mathrm{IE}| = \mathrm{pm}(G)^2 \end{aligned}$$

□

Whether or not a term $\operatorname{sign}(\pi)a(\pi)$, $\pi \in \mathrm{IE}$, is $+1$ or -1 depends on how we orient the graph $\vec{G}$. Ideally, we wish to choose an orientation in such a way that this is always[5] $+1$. Then we can compute the number of perfect matchings in G as $\operatorname{pm}(G) = \sqrt{\det(A_s(\vec{G}))}$. We will see that this is indeed possible for planar graphs.

To this end, recall that $\operatorname{sign}(\pi) = (-1)^{\text{number of even cycles in } \pi}$. For a cycle $c : i_1 \mapsto i_2 \mapsto \cdots i_k \mapsto i_1$ in a permutation π write

$$a[c] := a_{i_1 i_2} a_{i_2 i_3} \cdots a_{i_{k-1} i_k} a_{i_k i_1} .$$

Then, for $\pi \in \mathrm{IE}$,

$$\operatorname{sign}(\pi)a(\pi) = \prod_{c \text{ cycle in } \pi} (-a[c])$$

since in $\pi \in \mathrm{IE}$ all cycles are even, so the minus in "$(-a[c])$" takes nicely care of $\operatorname{sign}(\pi)$'s contribution. Note right away that for a cycle $c : i \mapsto j \mapsto i$ of length 2, $a[c] = a_{ij}a_{ji} = -1$ (skew-symmetry!) and therefore the term $(-a[c])$ is $+1$.

It is clear now what we want. Choose the orientation in such a way that for any cycle c in a permutation in IE, we have $a[c] = -1$. This will be the case, when we run along the cycle c in the graph $\vec{G}$ and the number of times we move against an edge of $\vec{G}$ is odd. The goal is set, we understand what needs to be done, so let us proceed.

Nice Cycles in G and Oddly Oriented Cycles in $\vec{G}$. We have to define some terminology for our next step. Along what we have just argued, both notions should appear naturally.

Let C be an undirected cycle in G of even length. We call C *oddly oriented* in $\vec{G}$ if going through C in some direction, we encounter an odd number of edges in $\vec{G}$ oriented in our traversal direction – and, therefore, also an odd number of edges oriented in the opposite direction (hence, since C is even, the definition does not depend on the direction in which we chose to traverse C).

[5] You correctly might want to argue that "always -1" would be just as good. However, the determinant of a skew-symmetric matrix is never negative, so "always -1" will not be possible.

For C a cycle in G with vertex set V_C, we call C *nice*, if $G[V_G \setminus V_C]$ (the subgraph of G induced by $V_G \setminus V_C$) has a perfect matching. Observe that if G has any perfect matching, then n has to be even (what we assumed) and every nice cycle has to be even. Note also, that if a permutation $\pi \in IE$ has a cycle c of length at least 4, then its undirected counterpart C is a nice cycle in G. And $a[c] = -1$ iff C is oddly oriented in $\vec{G}$.

So "nice" and "oddly oriented" let us nicely express a sufficient condition for what we want (it is actually also necessary, but we don't allow such distractions at this point).

Lemma 5.5. *If every nice cycle in G is oddly oriented in $\vec{G}$, then*

$$\det(A_s(\vec{G})) = \mathrm{pm}(G)^2 \ .$$

Orientations $\vec{G}$ with $\det(A_s(\vec{G})) = \mathrm{pm}(G)^2$ are called *Pfaffian*[6][7]. There are examples of graphs which do not allow a Pfaffian orientation, so we will restrict to a special class of graphs.

Pfaffian Orientations of Planar Graphs (Kasteleyn, 1967).[8] *Planar graphs* are graphs that can be drawn in the plane so that no pair of edges crosses. A concrete crossing-free embedding is called a *plane graph*. If v is the number of vertices of a connected plane graph, e the number of edges, and f the number of faces, then Euler's relation says that $v - e + f = 2$ (don't forget to count the unbounded face).

A maximal planar graph (i.e. no further edge can be added without violating planarity) is called a *triangulation*. In any crossing-free embedding of a triangulation, all faces (including the unbounded one) are triangles, at least as long as the number of vertices is at least 3.

Lemma 5.6. *Let $\vec{T}$ be a plane oriented triangulation with at least 3 vertices where every finite face (a triangle) has an odd number of edges oriented clockwise.*

- *Let C be an undirected cycle in T with k of its edges oriented clockwise in $\vec{T}$ and with v vertices of T inside C (i.e. in the region surrounded by C, not including C). Then $k \equiv v + 1 \pmod 2$.*

[6] This term comes out of the blue here, I know ... google, or even better, consult a Linear Algebra book!

[7] Johann Friedrich Pfaff, 1765-1825, mathematician born in Stuttgart, Carl Friedrich Gauss's supervisor.

[8] Piet Kasteleyn, 1924-1996, theoretical physicist born in Leiden, Netherlands.

- $\vec{T}$ *is a Pfaffian orientation.*

Proof. Let C be a cycle in T with f faces inside C. Let k_i, $i \in \{1..f\}$, be the number of edges of the ith face with clockwise orientation (an edge is oriented clockwise for one incident face and counter-clockwise for the other). Each of these k_i is an odd number. Hence

$$f \equiv \sum_{i=1}^{f} k_i \pmod 2 . \tag{5.1}$$

Let e be the number of edges inside C. Then by Euler's formula

$$\begin{aligned} & (v + |C|) - (e + |C|) + (f + 1) = 2 \\ \Rightarrow \quad & v - e + f = 1 \end{aligned} \tag{5.2}$$

and, moreover,

$$\sum_{i=1}^{f} k_i = k + e \tag{5.3}$$

since every edge inside C is oriented in clockwise direction on exactly one side. The ingredients are ready to cook.

$$k \underbrace{=}_{(5.3)} (\sum_{i=1}^{f} k_i) - e \underbrace{\equiv}_{(5.1)} f - e \underbrace{=}_{(5.2)} 1 - v \equiv v + 1 \pmod 2$$

For the fact that $\vec{T}$ is Pfaffian, observe that a nice cycle must have an even number of vertices inside (a perfect matching cannot go across the cycle). Therefore, every nice cycle has an odd number of edges oriented clockwise in $\vec{T}$, thus it is oddly oriented. □

Theorem 5.7 (Kasteleyn). *Every planar graph has a Pfaffian orientation which can be computed in linear time.*

Proof. Observe that if a graph G has a Pfaffian orientation $\vec{G}$, then all subgraphs of $\vec{G}$ (with some edges removed) are Pfaffian orientations. This holds, since removing edges just means setting some entries in $A_s(\vec{G})$ to 0. In this way we cannot generate negative terms in the sum $\sum_{\pi \in \mathrm{IE}} \mathrm{sign}(\pi) a(\pi)$.

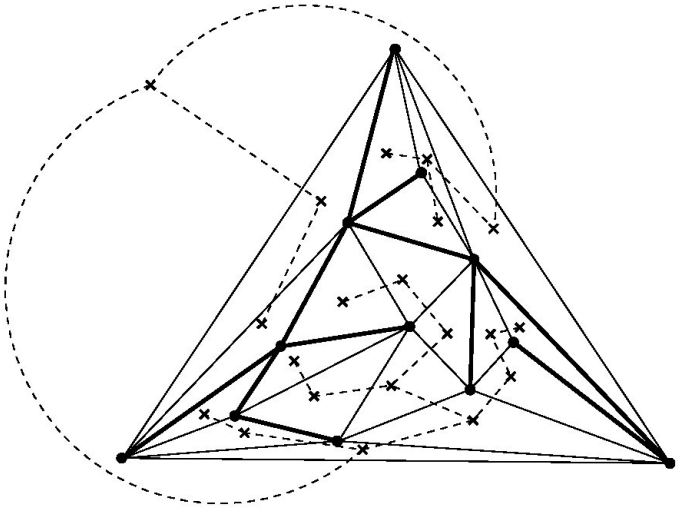


Figure 5.3: The trees B and B* for a triangulation.

So it suffices to show the claim for triangulations. Let us fix a crossing-free embedding of a triangulation T. T has a spanning tree B, so far so good. Now consider the dual graph B* of T where we use only edges not in B. That is, every face represents a vertex (including the infinite face), and two such vertices are connected by an edge if their faces share a common edge which is not in B. Since B has no cycles, B* is connected, and since B is connected, B* cannot have a cycle. So B* is a spanning tree[9] of the dual graph of T.

Towards the Pfaffian orientation $\vec{T}$, start by orienting the edges of B arbitrarily. Now consider a leaf in B*. This represents a face t, a triangle, where two edges are already oriented and one is still floating. Choose an orientation for this floating edge so that things go well for t, i.e. so that t has an odd number of clockwise oriented edges. Remove the vertex representing t from B*, and continue the same, but never choosing the infinite face, even if it turns into a leaf while B* gets truncated. In this way all faces have indeed an odd number of clockwise edges, except for the infinite face, which we are perfectly happy to accept.

The operations necessary for a planar graph – extending to a triangulation, computing a spanning tree and the dual tree, and then the truncation process – can be done in linear time (all easy, except for the triangulation step which requires basically a planarity test, which is not that obvious to

[9]It has $2n-4$ vertices if T, and thus B, has n vertices, if you ask

be done in linear time). □

Corollary 5.8. *The number of perfect matchings in a planar graph can be determined in polynomial time.*

Wrap-up, Upper Bound for the Number of Perfect Matchings in Planar Graphs. Hadamard tells us how big a determinant can be in terms of its entries.

Lemma 5.9 (Hadamard). *If $a_1, a_2, \ldots a_n$ are the columns of a matrix $A = (a_{ij})_{i,j=1}^n \in \mathbf{R}^{n\times n}$, then*

$$|\det(A)| \leq \prod_{i=1}^{n} \|a_i\| \leq \sqrt{\frac{1}{n}\sum_{ij} a_{ij}^2}^{n} .$$

($\|b\| = \sqrt{b_1^2 + b_2^2 + \cdots + b_k^2}$ for $b = (b_1, b_2, \ldots, b_k)$.)

The first inequality in Lemma 5.9 is usually called the Hadamard inequality and the latter inequality follows from it by using that the geometric mean is upper bounded by the arithmetic mean. An upper bound for the number of perfect matchings readily follows for planar graphs.

Theorem 5.10. *The number of perfect matchings in a planar graph with n vertices is at most $\sqrt[4]{6}^n$.*

Proof. The number of perfect matchings cannot go down if we add edges to a graph, therefore it suffices to consider triangulations for the upper bound. Such a triangulation T has $3n-6$ edges (provided $n \geq 3$), therefore the matrix $A := A_s(\vec{T})$ satisfies $\sum_{ij} a_{ij}^2 = 6n - 12 \leq 6n$. If $\vec{T}$ is Pfaffian then

$$\mathrm{pm}(T) = \sqrt{\det(A)} \leq \sqrt{\sqrt{6}}^n .$$

□

Exercise 5.11. Vertex Swaps in Orientations. *Let $\vec{G}$ be an orientation of a graph G and v a vertex in G. A swap at v in $\vec{G}$ changes the orientation of all the edges incident to v; we obtain a new orientation $\vec{G}'$ of G. Show that an even cycle is oddly oriented in $\vec{G}$ iff it is oddly oriented in $\vec{G}'$.*

Exercise 5.12. Prescribing Orientations.
Let G be a connected graph with a Pfaffian orientation and let $\vec{B}$ be a spanning tree of G with its edges oriented. Show that there is a Pfaffian orientation $\vec{G}$ where the edges in B are oriented as prescribed in $\vec{B}$.

Exercise 5.13. No Good Orientation.
(1) Show that a graph containing a complete bipartite $K_{2,3}$ cannot be oriented in such a way that every even cycle is oddly oriented. (Note that this includes planar graphs.)
(2) Show that $K_{3,3}$ cannot be oriented in such a way that all nice cycles are oddly oriented.

Exercise 5.14. Characterizing Pfaffian.
Show that $\vec{G}$ is Pfaffian iff every nice cycle is oddly oriented. (That is, supply the other direction of Lemma 5.5.)

Exercise 5.15. Swap-Equivalence.
Two orientations of a graph are called swap-equivalent, *if one can be obtained from the other by vertex swaps (see Exercise 5.11). Are any two orientations of a plane triangulation, where every finite face has an odd number of clockwise edges, swap-equivalent?*

Exercise 5.16. Hamiltonian Cycles in Planar Graphs.
Show that a planar graph with n vertices can have at most $\sqrt[4]{30}^n$ Hamiltonian cycles. (Perhaps start with a bound of $\sqrt{6}^n$, that should be easy.)

Exercise 5.17. Determinant Vanishes.
Show that the determinant of a skew symmetric matrix $A \in \mathbf{R}^{n \times n}$ is 0 if n is odd.