CS 270 Algorithms **Spring 2021**

Lecture 24: Perfect Matchings: polyhedral and algebraic techniques

Lecturer: Prasad Raghavendra Scribe: Julius Vering, Nate Young, Daniel Jing

In this lecture, we want to explore several techniques to solve the classical perfect matching problem. The matching problem can be considered a ‘cornerstone’ in the field of algorithms, as it is one of the earliest combinatorial problems for which algorithms were designed, including the Hungarian algorithm and algorithms based on augmenting paths. In fact, these algorithms were conceived before the notion of ‘efficiency’ of computation was firmly established via asymptotic runtime analysis and the complexity class P. In turn, these efficiency measures later justified the merit of the early algorithms compared to solving the matching problem via brute force.

In the previous lecture, we already talked about the bipartite matching problem and explored polyhedral techniques to solve the problem. Namely, we constructed a linear program for the bipartite matching problem and then argued that its polytope is integral, allowing us to conclude that solving the linear program is equivalent to solving the bipartite matching problem. In this lecture, we want to explore similar techniques for the more general perfect matching problem. Namely, we will first build upon the discussion from last lecture while discussing polyhedral techniques for solving the perfect matching problem. Then, we will explore an algebraic approach to solving both the bipartite and the general perfect matching problem. Finally, we will see the related problem of red-black matchings as an illustration of the power of the algebraic technique.

24.1 The polyhedral technique

Recall the perfect matching problem; given some graph $G = (V, E)$, we would like to find a subset of edges such that all the vertices are ‘paired up’ by being connected to exactly one other vertex via an edge in that subset. Put differently, the subset of edges is such that every vertex is incident on exactly one of the edges in the subset and no pair of edges share a vertex.

We would like to construct a linear program for the set of perfect matchings of G – that is, we would like to construct a polytope all of whose extremal points (“corners,” where in m dimensions there are m tight constraints) are integral, and therefore correct perfect matchings.

Consider the “obvious” polytope for perfect matching on a graph $G = (V, E)$, with the variables x_e for each edge $e \in E$. The intent of the variables is

$$x_e = \begin{cases} 1 \text{ if } e \in \text{matching} \\ 0 \text{ otherwise} \end{cases}$$

The constraints are then given as

$$\forall e \in E, \quad 0 \le x_e \le 1 \tag{24.1}$$

$$\forall v \in V, \quad \sum_{e \in \delta(v)} x_e = 1 \ , \tag{24.2}$$

where $\delta(v)$ is the set of edges incident on the vertex v.

Clearly, when the variables take integer values, this is correct, but unfortunately not all extremal points of this polytope are integral. Consider assigning value $\frac{1}{2}$ to all variables in the triangle graph in Figure 24.1. It is easy to check that this is indeed an extremal point of the polytope above, but it is clearly not integral.

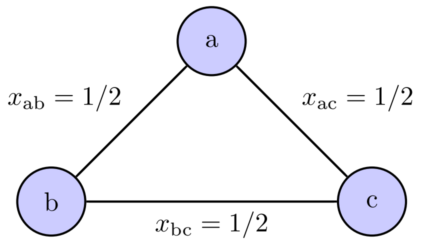


Figure 24.1: Assigning the value 1/2 to every edge in this graph satisfies both constraints 24.1 and 24.2 of the linear program and is, in fact, an extremal point, but it is clear that this assignment does not correspond to a perfect matching.

In order to fix this, we will introduce some more constraints, called the "odd-set constraints". They are derived from graph theoretic theorems; for the case of bipartite graphs, this is given by Hall's marriage theorem, stating that a bipartite graph has a matching if and only if it fulfills the so-called marriage condition [HV50]. A generalization for perfect matchings in general graphs is given by Tutte's theorem [PL86], itself a special case of the Tutte-Berge formula. From this theorem, we can derive the following constraint:

$$\forall\, U \subseteq V \text{ with } |U| \text{ odd}, \quad \sum_{e \in \delta(U)} x_e \geq 1 \ ,$$

where $\delta(U)$ is the set of edges with exactly one endpoint in U. That is, all sets of vertices of odd size must have at least one edge leading out of them. Note that this introduces some redundancy for subsets of size 1 with the constraints on each vertex from the naive polytope. The intuition for this condition is that, clearly, there cannot be a perfect matching on an odd number of vertices, so there must be some edge leaving the set if G has a perfect matching. Note that the bad assignment to the triangle graph from above is excluded by the odd-set constraint on the whole graph. Further, observe that there are an exponential number of these odd-set constraints, but a polynomial-time separation oracle still exists to find a violated odd-set constraint if there is one.

We will prove that this new polytope is exactly the convex hull of perfect matchings. This will imply that the extremal points of the polytope are integral, as the extremal points of the convex hull of perfect matchings are exactly the perfect matchings and therefore integral.

Theorem 24.1. *The polytope described above, including the odd-set constraints, is exactly the convex hull of the perfect matchings of G.*

Proof. First we will note that all perfect matchings are feasible in the polytope described; therefore the convex hull of perfect matchings is contained in our polytope. So, it will suffice to prove that our polytope is contained in the convex hull of matchings. However, there is a point in our polytope outside the convex hull of perfect matchings only if one such point is an extremal point of the polytope; so it will actually suffice to prove that all extremal points of our polytope are convex combinations of perfect matchings.

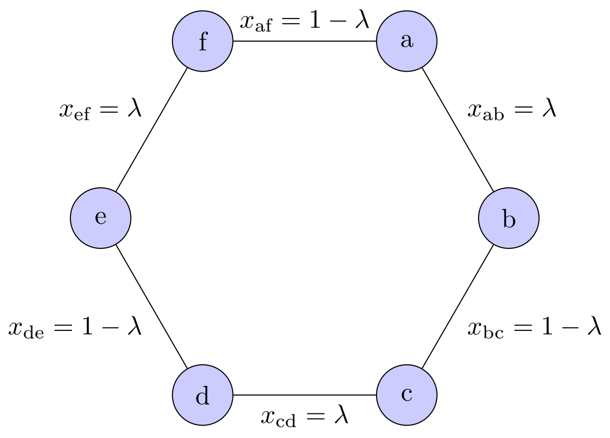
We will proceed by contradiction. Consider the smallest counterexample; that is, the smallest graph

$G = (V, E)$, measured by the sum $|V| + |E|$, for which our polytope has an extremal point which is not a convex combination of perfect matchings, and call the settings of the variables at this extremal point x.

Note the following fact about x: for all edges $e \in E$, $x_e \neq 0$ and $x_e \neq 1$. Otherwise, e could be removed from G entirely (including its endpoints in the case of $x_e = 1$, or leaving its endpoints in the case of $x_e = 0$) while leaving the rest of x unchanged to recover a smaller counterexample, which contradicts our assumption.

In order for the above to hold, note that it must be that $\deg(v) \geq 2$ for all vertices $v \in V$. Otherwise, if $v \in V$ has degree 1, then its one adjacent edge must have x and we can again remove both from the graph to construct a smaller counterexample. This implies $|E| \geq |V|$, as $\sum_{v \in V} \deg(v) = 2|E|$.

In the case where $|V| = |E|$, we can observe that $\deg(v) = 2$ for all $v \in V$ so our graph must be a collection of cycles. In fact, G must be a single cycle; if G had at least two disjoint cycles, we could consider any of the cycles as a smaller counterexample. Furthermore, as there is an odd-set constraint for the whole graph, the number of vertices in the graph must be even, so the cycle is even too. Note that any even cycle has exactly two distinct perfect matchings, so it is straightforward to observe that any feasible point in the polytope is a convex combination of the two perfect matchings, compare Figure 24.2. So, for the rest of the proof, we will assume that $|E| > |V|$.



$x_{\text{af}} = 1 - \lambda$
f
a
$x_{\text{ef}} = \lambda$
$x_{\text{ab}} = \lambda$
e
b
$x_{\text{de}} = 1 - \lambda$
$x_{\text{bc}} = 1 - \lambda$
d
c
$x_{\text{cd}} = \lambda$

Figure 24.2: An even cycle with the corresponding convex combination of the two possible perfect matchings. Note that all feasible points will be of this form: Suppose we assign the value λ to x_{ab}. Then, we must assign x_{bc} and x_{af} to $1 - \lambda$. Continuing, we find that the assingment above is the only possible solution.

In this case, since there are only $|V| < |E|$ matched-vertex constraints for $|E|$ variables, at least one of the constraints which is tight at our extremal point must be an odd-set constraint for an odd set with at least 3 vertices[1] (none of the 0-1 constraints may be tight, by the fact above). This constraint splits the graph into two sets $U, V \setminus U$, each of odd size; the sum of the x_e values assigned to the edges across this cut is exactly 1.

We will consider contracting together every vertex on the right side of the cut to form G_1, and separately every vertex on the left side to form G_2. If contracting the vertices creates self-loops, we delete them; if multi-edges are created, we keep them. Note that we may assume the original graph G to contain multi-edges already, the theorem holds just as well for that case. Compare Figure 24.3 for a visual representation of this process. Since the sum of the x_e assignments for e crossing the cut is 1, the contracted versions of the

[1]Note that the fact that $3 \leq |U| \leq |V| - 3$ follows as the odd-set constraints for sets of size 1 are already implied from the matched-vertex constraints.

assignment x, which we will denote as $x^{(1)}$ and $x^{(2)}$, satisfy all constraints of the polytope for their respective graph. Since both G_1 and G_2 are smaller than G, the assignments $x^{(1)}$ and $x^{(2)}$ are convex combinations of perfect matchings for G_1 and G_2, respectively, by the hypothesis.

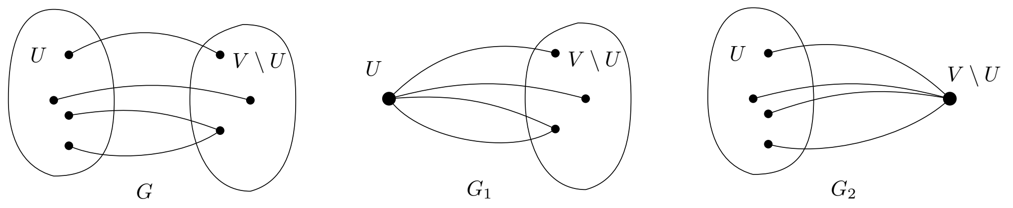


Figure 24.3: The construction of the graphs G_1 and G_2 from the original graph G. Note that the sum of the variables for the edges crossing the cut $(U, V \setminus U)$ is exactly 1 by our assumption.

As we argued above, for both $x^{(1)}$ and $x^{(2)}$, the sum over all $x_e^{(1)}$ and $x_e^{(2)}$, respectively, where e connects the contracted vertex to the rest of the graph, is exactly 1. From another angle, for any perfect matching M_1 in G_1 and any perfect matching G_2 in M_2, we know that there must be exactly one edge crossing the cut, so in particular, this holds for all the matchings that $x^{(1)}$ and $x^{(2)}$ are convex combinations of.

Furthermore, observe that for e connecting the contracted vertex to the rest of the graph, $x_e^{(1)} = x_e^{(2)} = x_e$ by construction. Hence, we observe that a x_e fraction of matchings in G_1 use the edge e and a x_e fraction of matchings in G_2 use the edge e. Importantly, all the matchings of G_1 using edge e also find a matching in the remainder of G_1 and all the matchings of G_2 using edge e find a matching in the remainder of G_2. Hence, we can "stitch" them together. This "stitching" process, when performed on every edge across the cut, will produce a convex combination of matchings on the entire graph G agreeing with x, which gives us our result. $\square$

Before we move on to explore a different technique for testing whether a graph has a perfect matching, we remark that the linear program that we derived in this section is of exponential size, but that a separation oracle can be constructed. The situation hence resembles the one in our discussion about spanning trees in the previous lecture. However, while there exists a polynomial-sized linear program for the spanning tree polytope, in a more recent result, Rothvoss showed that this is not the case for the perfect matching polytope [Rot17]. In other words, we need the separation oracle for the perfect matching linear program.

24.2 The algebraic technique for bipartite matching

In this section, we want to explore a new approach to solving the perfect matching problem. We will show a probabilistic algorithm that will rely on the properties of low-degree polynomials and the structure of the determinant of a matrix. To introduce the approach, we return to the bipartite matching problem that we considered last lecture. Later, we will see how to adapt the approach to the perfect matching problem.

Let $G = (V_1 \cup V_2, E)$ be a bipartite graph. Without loss of generality, we consider the case where $|V_1| = |V_2| = n$. We would like to know whether this graph has a perfect matching. To this end, we consider an $n \times n$ matrix B of variables that is defined as follows:

$$B_{ij} = \begin{cases} x_{ij} & \text{if } (i,j) \in E, \\ 0 & \text{otherwise} \end{cases}, \tag{24.3}$$

where each of the x_{ij} is a variable and $i \in \{1, \ldots, n\}$ corresponds to a vertex in V_1 and $j \in \{1, \ldots, n\}$ corresponds to a vertex of V_2. By definition, all edges in G are of this form, as G is bipartite. The matrix B is also called the Edmonds matrix. Since B is a matrix of variables, its determinant $\det(B)$ is simply a polynomial in the variables $\{x_{ij}\}_{(i,j)\in E}$. In particular, the Leibniz formula for the determinant of B is

$$\det(B) = \sum_{\pi \in S_n} \operatorname{sgn}(\pi) \prod_{i=1}^{n} B_{i\pi(i)} \ ,$$

where S_n denotes the finite symmetric group over n elements, i.e. the set of all permutations of the elements $\{1, \ldots, n\}$, and $\operatorname{sgn}(\pi)$ denotes the sign of the permutation π. The latter is defined as $\operatorname{sgn}(\pi) = (-1)^{N(\pi)}$, where $N(\pi)$ is the number of inversions in π, i.e. the number of pairs $i, j \in \{1, \ldots, n\}$ such that $i < j$ but $\pi(i) > \pi(j)$ or, intuitively, the number of swaps to go from sorted order to π. By the construction of B,

$$\prod_{i=1}^{n} B_{i\pi(i)} \neq 0 \ ,$$

only if $(i, \pi(i)) \in G$ for all $i \in \{1, \ldots, n\}$. If this is the case, then each vertex in V_1 is mapped to a unique vertex in V_2 such that the two vertices share an edge, i.e. for all $i \in V_1$, $(i, \pi(i)) \in E$ and each i corresponds to a unique $\pi(i)$ by definition. In other words, the above product is nonzero if and only if π is a perfect matching. Hence, we see that

$$\det(B) = \sum_{\pi \in S_n} \operatorname{sgn}(\pi) \prod_{i=1}^{n} B_{i\pi(i)} = \sum_{\substack{\pi \in S_n \\ \pi \text{ is a perfect matching}}} \operatorname{sgn}(\pi) \prod_{i=1}^{n} x_{i\pi(i)} \ .$$

For example, for the graph in Figure 24.4, we observe that there is only one matching and we can find that

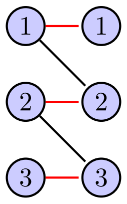


Figure 24.4: This graph has only one matching, given by the set of edges $\{(1, 1), (2, 2), (3, 3)\}$, colored in red. The corresponding permutation π is simply the identity permutation. Hence, we find that $\det(B) = x_{11} \cdot x_{22} \cdot x_{33}$.

therefore $\det(B) = x_{11} \cdot x_{22} \cdot x_{33}$. We conclude that G has a perfect matching if and only if

$$\det(B) = \sum_{\pi \in S_n} \operatorname{sgn}(\pi) \prod_{i=1}^{n} B_{i\pi(i)} \neq 0 \ .$$

It would be convenient if we could simply write out the determinant of the matrix and check whether it is the zero polynomial. However, we know that $|S_n| = n!$ and so the polynomial may have exponentially many terms in the worst case, making it impossible to fully write out this polynomial in polynomial time. Instead, we consider a simple randomized algorithm. Let S be some subset of a field such that $|S| \geq n^3$. This yields Algorithm 1.

The runtime of this algorithm is dominated by the time it takes to compute the determinant of the matrix. Since the matrix no longer consists of indeterminates but rather fixed entries that either equal zero or elements

Algorithm 1 Randomized algorithm for existence of a bipartite matching

```
1: procedure HasMatching($G = (V_1 \cup V_2, E), S$)
2:     Let $B$ be the matrix of variables from Equation (24.3).
3:     for $(i, j) \in E$ do
4:         Set $B_{ij} = x_{ij}$ to a randomly sampled element of $S$.
5:     end for
6:     Compute $\det(B)[\{x_{ij}\}]$.
7:     Output Yes if and only if $\det(B)[\{x_{ij}\}] \neq 0$. Otherwise, output No.
8: end procedure
```

of S, the determinant can be computed much faster. Naively, we can do so in $O(n^3)$ time using Gaussian elimination. In fact, one can prove that computing the determinant is as fast as matrix multiplication, so the algorithm above can run in time $O(n^\omega)$.

To show that the algorithm above is correct, at least with reasonable probability, we first observe that it always returns the correct answer when G does not have a perfect matching. If G does have a perfect matching, the algorithm falsely outputs No if and only if the polynomial $\det(B)$ vanishes on the sampled values $\{x_{ij}\}$. To bound the probability of this event, we introduce the following Lemma.

Lemma 24.2 (Schwartz-Zippel Lemma)**.** *Let P be a non-zero polynomial over the field $\mathbb{F}$, i.e. $P \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ with total degree $\deg(P)$, where the total degree is the highest degree of a monomial in P. Let S be a finite subset of $\mathbb{F}$ and let $x_1, \ldots, x_n$ be random elements of S. Then,*

$$\Pr[P(x_1, \ldots, x_n) = 0] \leq \frac{\deg(P)}{|S|} .$$

Proof. The proof proceeds by induction on n, the number of variables. For the base case, $n = 1$, we are considering a univariate polynomial of degree at most d. Here, we know that the polynomial has at most d roots in the field. Thus, the claim follows immediately, as we pick one of its roots out of the variables in S with probability at most $d/|S|$. It is clear that, otherwise, the polynomial is not equal to 0.

Now, suppose the Lemma holds for any polynomial on $n - 1$ variables. Then, we can write the polynomial P as

$$P(x_1, \ldots, x_n) = \sum_{i=1}^{\deg(P)} x_n^i P_i(x_1, \ldots, x_{n-1}) .$$

We assumed that P was a non-zero polynomial, so there must be at least one P_i that is non-zero. Let j be the largest index such that P_j is non-zero. Since the total degree of P is $\deg(P)$, we have that $\deg(P_j) = \deg(P) - j$. Hence, on randomly sampled $x_1, \ldots, x_{n-1}$ from S, we have that P_j is zero on these variables with probability at most $\frac{\deg(P)-j}{|S|}$. Then, if P_j is not zero at this assignment, we are left with a univariate polynomial in x_n of degree j. Now, if we randomly sample x_n from S, the resulting polynomial disappears with probability at most $j/|S|$ by the base case. Otherwise, the polynomial on the resulting assignment $x_1, \ldots, x_{n-1}, x_n$ is non-zero. By a union bound, we conclude that hence, the polynomial disappears with probability at most $\frac{\deg(P)-j}{|S|} + \frac{j}{|S|} = \frac{\deg(P)}{|S|}$, which proves the claim. □

Using the Schwartz-Zippel Lemma, we can lower bound the probability that $\det(B)$ vanishes on the sampled values by $\frac{n}{n^3} = \frac{1}{n^2}$ since the determinant for an $n \times n$ matrix is a polynomial of total degree at most n. Hence, the probability that Algorithm 1 outputs YES when G has a perfect matching is at least $1 - \frac{1}{n^2}$, although we can of course shrink this probability arbitrarily by sequential repetition.

Note that this algorithm only detects whether or not G has a matching. To get an algorithm that finds a matching, we can apply a standard reduction from search to detection; for every edge, we can remove it and use the above algorithm to check whether the resulting graph still has a matching. If it does, then we proceed on the graph with the removed edge; otherwise, we conclude that this edge is contained in the matching and put it back in the graph. Showing the correctness of this algorithm requires a straightforward argument relying largely on the union bound. We leave this as an exercise to the reader.

Remark. The Schwartz-Zippel Lemma is extremely powerful and widely used beyond applications such as the one described in this lecture. One of the key applications is in coding theory; for example, the relative distance property of the Reed-Muller code follows directly from the Schwartz-Zippel Lemma.

24.3 The algebraic technique for perfect matching

In this section, we want to adapt the techniques introduced in the previous section to the case of general graphs. Hence, let $G = (V, E)$ be some general graph. We would like to determine whether it has a perfect matching. In analogy with the previous section, we introduce a matrix of variables T, defined as follows:

$$T_{ij} = \begin{cases} +x_{ij} & \text{if } i < j \text{ and } (i,j) \in E, \\ -x_{ij} & \text{if } i > j \text{ and } (i,j) \in E, \\ 0 & \text{otherwise} \end{cases}, \tag{24.4}$$

where each of the x_{ij} is a variable and i, j both correspond to vertices in V. This matrix is called the Tutte matrix. For this definition, we note that all the variables above the diagonal have positive coefficents while those below have negative coefficients. Based on intuition gained in the previous section, we may consider the following claim:

Claim 24.3. *The determinant* $\det(T)$ *is a non-zero polynomial if and only if* G *has a perfect matching.*

Proof. Comparing the matrix B with the matrix T, observe that the difference from the bipartite case is that the rows and the columns of T refer to the same set of nodes, rather than two disjoint sets, which makes the analysis a bit more subtle. First, let us assume that M is a perfect matching for G and show that this implies that $\det(T) \neq 0$. Recall from the previous section the Leibniz formula for the determinant:

$$\det(T) = \sum_{\pi \in S_n} \operatorname{sgn}(\pi) \prod_{i=1}^{n} T_{i\pi(i)}$$

Consider the permutation π such that for any $(i,j) \in M$, we have $\pi(i) = j$ and $\pi(j) = i$. Then, we see that

$$\prod_{i=1}^{n} T_{i\pi(i)} = \pm \prod_{(a,b) \in M} x_{ab}^2$$

where x_{ab} corresponds to the entries $T_{a\pi(a)}$ and $T_{b\pi(b)}$, up to its sign. Crucially, we can see that $\prod_{(a,b)\in M} x_{ab}^2$ is not produced by any other permutation. To see this, observe that for the term x_{ab}^2 to exist, we see that we must have $\pi(a) = b$ and $\pi(b) = a$. Since this holds for all $(a,b) \in M$, the claim holds. Hence, summing over all permutations, this monomial cannot be cancelled out. Thus, if a perfect matching exists in G, then $\det(T) \neq 0$.

Now, let us assume that $\det(T) \neq 0$ and show that this implies that a perfect matching exists in G. Using the Leibniz formula for the determinant, it is clear that there must exist some permutation π such that

$$\operatorname{sgn}(\pi) \prod_{i=1}^{n} T_{i\pi(i)} \neq 0$$

such that this term is not cancelled out by the other terms. If the term above is non-zero, then $(i, \pi(i)) \in E$ for all $i \in V$. This implies that π is a union of cycles. To see this, pick any vertex $i \in V$ and repeatedly apply π. Every time we apply π, we move to a new vertex via an edge. Since the permutation is a bijection and we can therefore never arrive at the same vertex twice in this walk, we must eventually return to the vertex i. Note that there may also be vertices $i \in V$ such that $\pi(\pi(i)) = i$, i.e. both vertices i and j for $(i, j) \in E$ are matched with each other. Since these vertices are already matched as in a perfect matching, we will not discuss this case further and only consider vertices that are part of an actual cycle.

Now, we want to show that π must have only even length cycles. For contradiction, suppose that π has some cycle of odd length. Then, let π' be the permutation derived from π by inverting the odd length cycle. Note that $\operatorname{sgn}(\pi) = \operatorname{sgn}(\pi')$. To see this, one can observe that the sign of a permutation is determined by the number of even and odd (disjoint) cycles it can be uniquely decomposed into, which clearly does not change if we simply invert one of these cycles[2]. Furthermore, for each vertex i in the inverted cycle, we have that $T_{i\pi(i)} = -T_{\pi(i)\pi'(\pi(i))} = -T_{\pi(i)i}$ because $\pi'(\pi(i)) = i$ and $T_{ij} = -T_{ji}$ for all $i \neq j$ by definition. In other words, for each vertex i in the inverted cycle, we replace $T_{i\pi(i)}$ in the product $\prod_{i=1}^{n} T_{i\pi(i)}$ by $T_{\pi(i)\pi'(\pi(i))} = T_{\pi(i)i} = -T_{i\pi(i)}$ to get the product $\prod_{i=1}^{n} T_{i\pi'(i)}$. Therefore, we 'flip' the sign of the product $\prod_{i=1}^{n} T_{i\pi(i)}$ once for each vertex in the inverted cycle and since we assumed it to be of odd length, we find that $\prod_{i=1}^{n} T_{i\pi(i)} = -\prod_{i=1}^{n} T_{i\pi'(i)}$. Therefore, $\operatorname{sgn}(\pi)\prod_{i=1}^{n} T_{i\pi(i)}$ and $\operatorname{sgn}(\pi')\prod_{i=1}^{n} T_{i\pi'(i)}$ add to zero, i.e. cancel each other out. However, since we assumed that the term is not cancelled out by any other terms, we conclude that π cannot have an odd length cycle.

Since π has only even length cycles, π corresponds to a matching. This is true because for any even length cycles, we can simply pair up each vertex in the cycle with one of the vertices next to it and convert the even length cycle into a matching. Hence, we conclude that G must indeed have a perfect matching, as we wanted to show. $\square$

Using this claim, it is clear how we can compute whether a perfect matching exists in the graph G. Namely, we simply run Algorithm 1. Some slight modifications are necessary and while they are straightforward, we include the modified version, Algorithm 2, below for completeness.

As in Algorithm 1, we let S be a subset of a field with $|S| \geq n^3$. The runtime of the proof again depends on the runtime of computing the determinant, which can be accomplished in $O(n^\omega)$ time. The correctness of the algorithm again follows from Lemma 24.2 as the degree of the polynomial and the subset size remain unchanged. Finally, we can again use this algorithm to construct an algorithm to find a perfect matching by using the same method that we described in Section 24.2.

At this point, we remark that the fundamental reason this algorithm and Algorithm 1 work is that we are not simply considering an arbitrary polynomial with total degree at most n and potentially exponentially many

[2]Showing this is an elementary exercise in group theory, which we will omit. Curious readers may consult any abstract algebra textbook or the Wikipedia page on the Parity of a Permutation.

Algorithm 2 Randomized algorithm for existence of a perfect matching

```
1: procedure HasMatching($G = (V, E), S$)
2:     Let $T$ be the matrix of variables from Equation (24.4).
3:     for $(i, j) \in E$ do
4:         Set $T_{ij} = x_{ij}$ to a randomly sampled element of $S$. Set $T_{ji}$ to $-x_{ij}$.
5:     end for
6:     Compute $\det(T)\,[\{x_{ij}\}]$.
7:     Output Yes if and only if $\det(T)\,[\{x_{ij}\}] \neq 0$. Otherwise, output No.
8: end procedure
```

terms. Rather, since the polynomial is the determinant of a matrix of indeterminates, it has a large amount of additional structure, which is what we take advantage of to be able to evaluate the polynomial so quickly, which may not be the case when considering a general polynomial of total degree n and exponentially many terms.

24.4 Red-black matchings

In this section, we want to illustrate the power of the algebraic technique from the previous section to derive an efficient randomized algorithm for a problem for which no deterministic algorithm is known. We consider the following problem: Given a bipartite graph $G = (V_1 \cup V_2, E)$ where each edge $e \in E$ is assigned to be either red or black and some integer $k < n$, we want to find a perfect matching for G with exactly k red edges, or report that no such matching exists. Since we are again considering a bipartite graph, we would hope that we could modify the algorithm described in Section 24.2 to solve this problem.

Let E_B denote the set of black edges and E_R denote the set of red edges, where $E_B \cup E_R = E$. We define a modified version of the Edmonds matrix B described in Section 24.2 as follows:

$$B_{ij} = \begin{cases} x_{ij} & \text{if } (i,j) \in E_B, \\ y \cdot x_{ij} & \text{if } (i,j) \in E_R, \\ 0 & \text{otherwise} \end{cases} , \tag{24.5}$$

where x_{ij} are again variables but we additionally introduce a new variable y. Let us consider the Leibniz formula for the determinant of the matrix B to find

$$\det(B) = \sum_{\pi \in S_n} \operatorname{sgn}(\pi) \prod_{i=1}^{n} B_{i\pi(i)} \ .$$

From our discussion in Section 24.2, we recall that if G has any perfect matching, there exists a permutation π that corresponds to that matching and the corresponding product in the Leibniz formula is not cancelled out. With the additional variable y, we can observe that if M is a matching in G, we have that $\det(B)$ contains a term of the form

$$y^{[\#\text{ of red edges}]} \cdot \prod_{(a,b) \in M} x_{ab} \ .$$

Since for any π that does not correspond to a perfect matching, there exists a vertex $i \in V_1$ such that $(i, \pi(i)) \notin E$ and thus $B_{i\pi(i)} = 0$, we can write

$$\det(B) = \sum_{j=0}^{n} y^j q_j(x) \ ,$$

where $q_j(x)$ is a polynomial corresponding to all matchings of G such that the matching contains j red edges. Since none of the terms corresponding to a matching cancel each other out, we conclude that G has a k-red matching if and only if the coefficient of y^k in $\det(B)$ is non-zero.

Clearly, the naive way of checking whether G has a k-red matching by computing the coefficient of y^k directly is not efficient, as computing the determinant of B may take exponential time, as described in Section 24.2. Hence, we again take the randomized approach. First, observe that if we fix x_{ij} to α_{ij} for all $(i,j) \in E$, $\det(B)[\{\alpha_{ij}\}, \cdot]$ is a univariate polynomial in y of degree at most n given by $\sum_{j=0}^{n} y^j q_j(\alpha)$. Thus, evaluating $\det(B)[\{\alpha_{ij}\}, \cdot]$ at $n+1$ values of y allows us to interpolate the coefficient $q_j(\alpha)$, so we can simply check whether $q_j(\alpha) \neq 0$. This yields Algorithm 3.

Algorithm 3 Randomized algorithm for existence of a red-black matching

```
1: procedure HasMatching($G = (V_1 \cup V_2, E), S$)
2:     Let $B$ be the matrix of variables from Equation (24.5).
3:     for $(i,j) \in E$ do
4:         Set $x_{ij}$ to a randomly sampled element of $S$. This gives $B_{ij} = x_{ij}$ if $(i,j) \in E_B$ and $B_{ij} = y \cdot x_{ij}$
   if $(i,j) \in E_R$.
5:     end for
6:     Compute $\det(B)\,[\{\alpha_{ij}\}, \cdot]$.
7:     Interpolate the coefficient $q_j(\alpha)$ of the monomial $y^k$.
8:     Output Yes if and only if $q_j(\alpha) \neq 0$. Otherwise, output No.
9: end procedure
```

As in Algorithm 1, we let S be a subset of a field with $|S| \geq n^3$. Note that interpolating the coefficients of the polynomial, even naively, can be done in $O(n^2)$. Hence, the runtime of this algorithm is $O(n^\omega)$, as it is again dominated by the time it takes to compute the determinant. Note that the polynomial interpolation is deterministic, so the correctness of the algorithm only depends on whether or not $q_j(x)$ vanishes on the samples $\{\alpha_{ij}\}$. Since $q_j(x)$ has total degree at most n, the correctness again follows from Lemma 24.2. As in the previous two sections, we can again construct an algorithm to find a k-red matching using the algorithm described above.

24.5 Polynomial identity testing

In the previous section, we showed a randomized algorithm for a problem for which no deterministic algorithm is known. This can be seen as a concrete case of a fundamental open question in complexity theory, namely whether all problems that have randomized algorithms have deterministic algorithms as well. Returning to the concrete case, we note that the questions we studied in the previous three sections are special cases of a problem known as polynomial identity testing. In this problem, we are given as our input a formula or circuit for a polynomial $P[x]$. Our goal is to see whether the polynomial is identically zero, i.e. $P[x] \equiv 0$, after writing out and perhaps simplifying the form given to us. For this well-defined algorithmic question, a simple randomized algorithm is exactly what we did above, namely plugging in random values. As the Schwartz-Zippel Lemma shows, if we sample from a large enough set, with high probability, the polynomial will not vanish unless it is identically zero.

No deterministic algorithm is known for this problem. Clearly, a deterministic algorithm for polynomial identity testing would give a deterministic algorithm for red-black matchings and would allow us to convert the algorithms described above into deterministic algorithms. Furthermore, any deterministic algorithm for this problem would be a major breakthrough in complexity theory and would have far-reaching consequences. Nevertheless, it is believed that a deterministic algorithm exists for this problem[3].

References

[HV50] Paul R. Halmos and Herbert E. Vaughan. The marriage problem. *American Journal of Mathematics*, 72(1):214–215, 1950.

[PL86] M.D. Plummer and L. Lovász. *Matching Theory*. ISSN. Elsevier Science, 1986.

[Rot17] Thomas Rothvoss. The matching polytope has exponential extension complexity, 2017.

[3]This question is known as derandomization. In particular, a widespread conjecture is the existence of strong pseudorandom number generators, which implies that $\mathsf{P} = \mathsf{BPP}$, i.e. all languages with randomized algorithms have efficient deterministic algorithms.