CS 759 : Perfect Matchings

Amit Rajaraman

Last updated February 4, 2023

Contents

§1. The Bipartite Setting

1.1. Basic matching theory

A well-known result characterizing bipartite graphs with perfect matchings is the following.

Theorem 1.1 (Hall's Theorem)**.** A bipartite graph $G(X, Y)$ has a perfect matching iff $|X| = |Y|$ and $|\Gamma(S)| \geq |S|$ for all $S \subseteq X$.

Corollary 1.2. For $d \geq 1$, any d-regular bipartite graph has a perfect matching. In particular, it is decomposable into d perfect matchings.

Henceforth, assume that G is a bipartite graph with $|X| = |Y| =: n$. Although it is easy through Hall's Theorem to figure out if G *has* a perfect matching, it is harder to count the number of perfect matchings. Indeed, counting turns out to be #¶-hard, and even counting modulo a prime p is hard. Any perfect matching of a bipartite graph may be thought of as a permutation $\sigma \in S_n$. Recall that any permutation π has a sign

$$\text{sign}(\pi) = (-1)^{\text{number of even cycles in } \pi}.$$

See Section 1.1 of the author's Combinatorics I notes for more details.

We may associate with G an $n \times n$ *bipartite adjacency matrix* M, where $M_{ij} = 1$ iff the ith vertex in X is adjacent to the jth vertex in Y. Note that

$$\det(M) = \sum_{\sigma \in S_n} \text{sign}(\sigma) \prod_{i=1}^{n} M_{i,\sigma(i)}.$$

Note that the product is nonzero (and in this case equal to 1) iff σ corresponds to a perfect matching of G. Consequently, if every perfect matching has the same sign, we can count the number of perfect matchings by merely looking at the absolute value of the determinant of the bipartite adjacency matrix. One interesting problem is to determine which graphs are such thata all perfect matchings have the same sign. We also have

$$\text{perm}(M) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} M_{i,\sigma(i)} = \text{number of perfect matchings},$$

but unlike the determinant, the permanent is hard to compute.

Let M_1, M_2 be perfect matchings of G. Note that $M_1 \triangle M_2$, the set comprising of the edges in precisely one of M_1, M_2, is a disjoint union of alternating cycles. Here, an alternating cycle means that the edges alternate being in M_1, M_2. M_2 has the same sign as M_1 iff there is an even number of cycles which have lengths divisible by 4.
We can use the above observation to check if all matchings have the same sign. Given a perfect matching M, all permutations have the same sign (as M) iff there exists no cycle of length divisible by 4 with edges alternating in M. How do we do this? First convert the bipartite graph to a directed one by assigning to each edge a direction from X to Y. The problem then boils down to determining if there exists an even directed cycle in the graph obtained by contracting the matching edges. This seems simple, but is in fact far from trivial. We know now due to Seymour [cite] that this is possible in polynomial time. We do not discuss the details in general.
Consider the specific case where G is 3-regular. The Heawood graph is known to have all matchings of the same sign.

Example 1 (Heawood graph)**.** The Heawood graph has vertex set $\mathbb{Z}_{14}$,

and we can use a construction procedure called the *star product* (sometimes called *splicing*) or to build up larger graphs with the same property. In fact, McCaig showed that the only 3-regular graphs with this property are obtained in this manner from the Heawood graph. There is also only one strongly 2-connected 2-regular directed graph without any even cycle, which is obtained by contracting the Heawood graph.

1.2. Pfaffians

Now, requiring that all matchings have the same sign is rather restrictive. A better idea might be to change the nonzero entries of the bipartite adjacency matrix in a way that ensures nonzeroness of the overall determinant iff there exists a perfect matching. More specifically, Polya asked if we can we change some of the 1 entries to -1 such that the resulting determinant becomes equal to the number of perfect matchings. For example, changing the bipartite adjacency matrix of $K_{2,2}$ to

$$\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$$

makes the determinant equal to 2, the number of perfect matchings. Another example is changing the bipartite adjacency matrix of $K_{3,3}$ with an edge removed to

$$\begin{pmatrix} -1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & 0 \end{pmatrix}.$$

That is, Polya's problem asks whether some of the 1s can be changed to -1s such that for any matching σ, $\operatorname{sign}(\sigma) = \prod_{i=1}^{n} a_{i\sigma(i)}$ (or possibly the negative for all matchings σ). That is, for any two matchings σ, π,

$$\operatorname{sign}(\sigma) \prod_{i=1}^{n} a_{i\sigma(i)} = \operatorname{sign}(\pi) \prod_{i=1}^{n} a_{i\pi(i)}.$$

We can think of the changed ± 1 entries as assigning directions to the edges in the graph, with $+1$ edges being directed from A to B and -1s from B to A (this is called an *orientation* of the graph).
Given a perfect matching M, say with positive sign, we would like to test if the matching obtained by switching the edges in any alternating cycle contributes the same way to the determinant as M. Let M_1 and M_2 be two matchings differing only on an alternating cycle, and let a_1, b_1 be the number of edges on the cycle in M_1 directed from A to B and B to A respectively, and a_2, b_2 the number of edges on the cycle in M_2 directed from A to B and B to A respectively. Let $\ell = a_1 + b_1 = a_2 + b_2$. The contribution by both matchings is the same iff

$$(-1)^{b_1} \operatorname{sign}(M_1) = (-1)^{b_2} \operatorname{sign}(M_2) = (-1)^{b_2}(-1)^{\ell+1} \operatorname{sign}(M_1) = (-1)^{a_2+1} \operatorname{sign}(M_1),$$

that is, $(-1)^{b_1+a_2} = -1$. In other words, there is an odd number of edges oriented along the direction of traversal of the cycle (this does not depend on the direction since the size of the alternating cycle is even).

Definition 1.3. Given an orientation of a graph, an even cycle (in the undirected graph) is said to be *oddly oriented* if there is an odd number of edges oriented along the direction of traversal.

Definition 1.4 (Pfaffian orientation)**.** A Pfaffian orientation is an orientation of the edges of a graph such that all alternating cycles are oddly oriented with respect to some perfect matching.

Note that in a bipartite graph, if a Pfaffian orientation exists for one perfect matching, it is Pfaffian for all perfect matchings.

Thus, if we can find a Pfaffian orientation of a given bipartite graph, we can determine the number of perfect matchings.
We may assume without loss of generality that if there exists a Pfaffian orientation, all edges in the matching we have are oriented from A to B – if not, we can reverse the direction of all edges incident on the A-vertex of the "wrongly" oriented edge.
We can reduce this problem to one on directed graphs. As before, reduce the original bipartite graph to a directed graph by contracting the edges in the matching, directing an edge that was originally between a_i and b_j from v_i to v_j. Then, the existence of a Pfaffian orientation is equivalent to the existence of a $0, 1$-weight assignment to the edges such that no (directed) cycle has even weight. If the Pfaffian orientation orients a a_ib_j edge from a_i to b_j, we assign weight 1 to the directed edge v_iv_j, and we assign weight 0 otherwise.
The above reduction can be used to easily prove that $K_{3,3}$ does *not* have a Pfaffian orientation. More generally, if the directed graph has a weak odd double cycle (an odd cycle with edges in both directions at all edges in the cycle), the graph does not have a Pfaffian orientation. Let us now make a couple of observations.

1. We may assume that no vertex has indegree and outdegree 1. Vertices with degree 2 (indegree and outdegree 1) can be removed by assigning the sum of the two edges' weights to the contracted edge.
2. We may assume that no vertex has neither indegree nor outdegree 1. If such a vertex exists, we can split it into two vertices, one with all the incoming edges, one with all the outgoing edges, and an edge from the former to the latter. If this middle edge is assigned weight 0 in a Pfaffian weighting, we trivially get a Pfaffian weighting for the original graph. If it is assigned weight 1, we can get a Pfaffian weighting with weight 0 by complimenting the weights of all edges.

Definition 1.5. A digraph is said to be *even* if in any assignment of $\{0, 1\}$-weights to the edges, some cycle has even weight.

Proposition 1.6. If the XOR of an odd number of cycles gives the empty set, one of these cycles must be even for any $\{0, 1\}$-assignment of weights.

This is also the reason why weak odd double cycles are even.
It is not immediately clear whether one can even check if a given orientation is a Pfaffian orientation. That is, given a directed graph with $\{0, 1\}$-weights, does it contain a cycle of even weight? This is a co-NP problem, since it can be proved to be false by giving a cycle of odd weight. The original problem asking whether a Pfaffian orientation exists asks whether there *exists* a weight assignment such that *for all* cycles, the cycle has odd weight – this is an example of a Σ_2 problem. Similarly, NP (which has a single $\exists$) is sometimes called Σ_1 and co-NP (which has a single $\forall$) is called Π_1.

First off, we shall show that determining if a given orientation is Pfaffian in polynomial time is equivalent to determining the existence of a Pfaffian orientation in polynomial time.

Proposition 1.7. Every strongly connected graph has an *ear decomposition*, that is, a partition of the edges into $C_0, P_1, \ldots, P_k$, where C_0 is a cycle and each P_i is a path whose endpoints are in $C_0 \cup P_1 \cup P_2 \cup \cdots \cup P_{i-1}$ and whose internal vertices (if any) are not contained in $C_0 \cup P_1 \cup P_2 \cup \cdots \cup P_{i-1}$.

Theorem 1.8. Consider the problem EVEN-DIGRAPH of determining if for a given digraph, every assignment of $\{0,1\}$-weights to the edges gives an even weight cycle. Consider the problem EVEN-CYCLE of determining if a given digraph has a cycle of even length. EVEN-DIGRAPH is in P iff EVEN-CYCLE is in P.

Proof. We first reduce EVEN-CYCLE to EVEN-DIGRAPH.
In EVEN-CYCLE, we may assume without loss of generality that the graph is strongly connected. Consider the ear decomposition $(C_0, P_1, \ldots, P_k)$ of this graph. Let G_i be the (strongly connected) graph formed by $C_0 \cup P_1 \cup \cdots \cup P_{i-1}$, and let C_i be any cycle in G_i that contains the ear P_i. It turns out that the (C_i) form a basis of all cycles in the underlying undirected graph (under XORing).
If some C_i is even, we are done since we have found an even cycle. Otherwise, run the algorithm to check if the graph is even. If yes, output yes, and output no otherwise. To prove this, it suffices to show that iff all C_i are odd and there exists an even cycle C, thenn the graph must be even. Indeed, because the C_i form a basis, we can write C as a XOR of an *even* number of C_i (because C is even and each C_i is odd). However, the XOR of this even number of cycles with C (an odd number of cycles overall) is empty, so we are done by Proposition 1.6.

Let us now reduce EVEN-DIGRAPH to EVEN-CYCLE.
We shall construct a weighting such that if there is an even cycle with respect to this weighting, there is an even cycle with respect to any weighting. If we manage to assign weights such that every basic C_i cycle has odd weight, we are done. Furthermore, it is easy to do this by inductively building up the graph ear-by-ear – assign a weight of 0 or 1 to a specific edge in P_i such that C_i is odd, and assign 0 to all other edges in the ear. ■

Theorem 1.9. Every planar graph has a Pfaffian orientation.

Proof. First off, we claim that every planar graph has an orientation such that for every internal face, there are an odd number of edges directed in the clockwise direction. This can be proved by induction on the number of edges. If the graph is a tree, orient it arbitrarily. Consider some edge that is incident on the external face. If removing this edge disconnects the graph (so the external face is the only face it is incident on), we can orient it arbitrarily since it is not part of any internal face. Otherwise, consider some such orientation for the graph minus the edge. Then, when adding this edge back, orient it in a way that ensures the internal face the edge is incident on has an odd number of clockwise edges. Note that this inductive proof also yields an algorithm to *find* such an orientation for any planar graph.
We claim that any such orientation is a Pfaffian orientation.
It may be seen using Euler's formula that the orientation is such that for any cycle, the number of edges in a clockwise direction will be of parity opposite to the number of vertices inside the cycle. If v is the number of vertices inside a length ℓ cycle, f is the number of faces inside the cycle, and e is the number of edges within the cycle, we have $(v+\ell) - (e+\ell) + f = v - e + f = 1$ (because the external edge is not being considered). Now note that if we traverse every internal face in a clockwise direction, every internal edge contributes in a clockwise manner to a cycle exactly once. Now let k be the number of edges on our ℓ-cycle which are clockwise. Then, $k+e$ is the sum of f odd numbers, so $k+e$ and f have the same parity. We also have by the use of Euler's formula that f and $v-e$ have opposite parities, so $k+e$ and $v-e$ have opposite parities, so k and v have opposite parities as desired.
However, this near-directly implies that the orientation is Pfaffian. Consider any alternating cycle, and note that no vertex within the cycle can be matched to a vertex outside the cycle, so there is an even number of vertices within the cycle and thus the alternating cycle is oddly oriented. ■

For bipartite planar graphs, since the above proof gives an algorithm to find a Pfaffian orientation, this implies that we can count the number of perfect matchings in polynomial time.
What is the significance of this result for general (non-bipartite) planar graphs? It turns out that even in general, we can construct a matrix using a Pfaffian orientation whose determinant is related to the number of perfect matchings.

This matrix A has

$$a_{ij} = \begin{cases} 1, & \text{edge } ij \text{ is directed from } i \text{ to } j, \\ -1, & \text{edge } ij \text{ is directed from } j \text{ to } i, \\ 0, & \text{there is no edge } ij. \end{cases}$$

We claim that this matrix has determinant equal to the square of the number of perfect matchings of the graph.

Definition 1.10 (Pfaffian)**.** Let M be a matching of K_{2n}. Suppose we order the matching M by writing down the edges in a specified order and the vertices within the edges – for example, the matching $\{\{1,2\},\{3,4\}\}$ could be written as $1,2,3,4$ or $1,2,4,3$. More generally, if the edges in M are $\{i_1j_1,\ldots,i_nj_n\}$, order it in some fixed manner as $i_1j_1\cdots i_nj_n = \sigma_M$, a permutation of $[2n]$. Let A be a $2n \times 2n$ skew-symmetric matrix. The *Pfaffian* of A is

$$\sum_{\substack{\text{all perfect matchings } M \text{ of } K_{2n} \\ \sigma_M = i_1j_1i_nj_n}} \operatorname{sign}(\sigma_M) \prod_{r\in[n]} A_{i_rj_r}.$$

Defining the above requires one to verify that we get the same quantity irrespective of how we order any matching M, but we omit the proof of this as it is relatively straightforward.
Finally, we must show that if we have a Pfaffian orientation and consider the matrix A defined before the above definition, it has Pfaffian equal to the number of perfect matchings of the graph (in absolute value).

Proposition 1.11. The determinant of a $2n \times 2n$ skew-symmetric matrix is equal to the square of the Pfaffian of the matrix.

The idea behind the proof is that two perfect matchings together can be thought of as a permutation on $[2n]$, with cycles in the graph being cycles of the permutation.

Proposition 1.12. If we have a Pfaffian orientation, then every perfect matching contributes the same ± 1 sign to the Pfaffian of the corresponding matrix. In particular, the number of perfect matchings of G is equal to the square root of the determinant.

Characterizing the class of graphs which have Pfaffian orientations is one of the largest open problems in matching theory. As of the time of writing, all we know is the Heawood graph, planar graphs, and graphs obtained by combining known graphs with Pfaffian orientations in some specified manner.

1.3. Red-blue matchings

Given a bipartite graph G with a perfect matching, suppose we color each edge as red or blue. Does there exist a perfect matching with some specified number k of red edges?
This problem is not known to be solvable in polynomial time as of the time of writing, but is also not known to be NP-complete.
In the scenario where the graph has a Pfaffian orientation, we can take the corresponding matrix and replace each red edge with a variable r (signed with the corresponding sign of the matrix depending on the Pfaffian orientation). The question then just asks whether there the r^k term has nonzero coefficient, which can be determined in polynomial time, either by finding the determinant as a polynomial, or using interpolation since we know that the polynomial has degree at most n.

The above Pfaffian case even counts the number of perfect matchings with k red edges, however. Is there an easier way if we want to just determine existence?
As in the usual perfect matching case, if we take the determinant of the adjacency matrix replacing all red edges with the variable r, if the coefficient of r^k in this polynomial is 0, then there is a matching with k red edges. We leverage this idea by assigning a random weight to each red edge; with good $(\Omega(1))$ probability, the resulting determinant is nonzero if there exists a matching with k red edges.

Theorem 1.13. Let G be a bipartite graph. For $e = ij \in E$, the weight w_e is uniformly randomly drawn from $[2m]$. Consider the matrix M defined by

$$M_{ij} = \begin{cases} 2^{w_{ij}}, & ij \in E \text{ and } ij \text{ is blue} \\ 2^{w_{ij}} r, & ij \in E \text{ and } ij \text{ is red}, \\ 0, & \text{otherwise.} \end{cases}$$

If G has no perfect matching with k red edges, then the coefficient of r^k in $\det(M)$ is 0. If G has as perfect matching with k red edges, the coefficient of r^k in $\det(M)$ is nonzero with probability at least $1/2$.

Note that the size of the bit representation of $2^{w_{ij}}$ is still polynomial in the input size n.

Proof. A matching M with k red edges contributes a term $\pm 2^{\sum_{e \in M} w_e} r^k$ to the determinant. That is, the coefficient of r^k in $\det M$ is

$$\sum_{\text{matchings } M \text{ with } k \text{ red edges}} \pm 2^{\sum_{e \in M} w_e}.$$

Suppose that our weight assignment is such that there is a unique minimum weight perfect matching with k red edges. In this case, the coefficient is nonzero! Indeed, the bit in the representation corresponding to this minimum weight matching cannot be cancelled out by any other matching. So, if we show that there is a unique minimum weight perfect matching with probability at least $1/2$, we are done. A stronger version of this is proved in the Isolation Lemma. ■

The following surprising result was proved by Mulmuley, Vazirani, and Vazirani.

Lemma 1.14 (Isolation Lemma)**.** Let E be a set of m elements and $\mathcal{S} \subseteq 2^E$ an arbitrary nonempty family of subsets of E. Independently and uniformly randomly assign to each element of E a weight in $\{1, 2, \ldots, N\}$. Then,

$$\Pr\left[\mathcal{S} \text{ has a minimum weight set}\right] \geq 1 - \frac{m}{N},$$

where the weight of a set is the sum of the weights of the elements in it.

We get the desired result in the context of Theorem 1.13 on setting E to be the set of edges, $\mathcal{S}$ to be the collection of perfect matchings with k red edges, and $N = 2m$.

Proof. Let $E = \{e_1, \ldots, e_m\}$. Split $\mathcal{S}$ into two parts $\mathcal{S}_0, \mathcal{S}_1$, where $\mathcal{S}_0 = \{T \in \mathcal{S} : e_1 \in T\}$ and $\mathcal{S}_1 = \mathcal{S} \setminus \mathcal{S}_0$. Let us look at the event E that there is a minimum weight set that contains e_1 and a minimum weight set that does not contain e_1. This means that the minimum weight set in $\mathcal{S}_0, \mathcal{S}_1$ are equal.
What happens if we fix the weights of all elements other than e_1? The minimum weight in $\mathcal{S}_1$ is determined, and the minimum weight in $\mathcal{S}$, is just equal to some fixed quantity plus the weight of e_1. In particular, there is at most one

value of $w(e_1)$ such that the two minimum weights are equal. Therefore, $\Pr[E] \leq 1/N$. In general, taking the union bound, we have

$$\Pr[\text{there exist two min wt sets}] = \Pr\left[\bigcup_{i\in[m]} \text{there exist min wt sets containing } e_i \text{ and not containing } e_i\right] \leq \frac{m}{N}. \quad \blacksquare$$

In fact, the above is known to be true with the larger probability $\left(1 - \frac{1}{N}\right)^m$.
The fact that this problem has a randomized polynomial time algorithm is good reason to believe that it is not NP-complete; no NP-complete problem is known to have a randomized polynomial time algorithm.
Consider three graph objects which are specific subsets of edges – spanning trees, paths from u to v, and perfect matchings.

- Determining existence of any of these three is in P.
- Next, we have the search versions of these problems of determining an object of the required form: for the first two, the proof of existence also gives an algorithm for finding an object. For the third, we can adapt the Mulmuley-Vazirani-Vazirani determinant idea used to determine the existence of a perfect matching to also find a perfect matching; we omit the details. Regardless, all three can again be done in polynomial time.
- Next, suppose we have to find the minimum weight object. This problem is in P for spanning trees. For paths, it can be done in polynomial when all weights are non-negative, and it is NP-complete otherwise. The problem is in P for perfect matchings.
- Next, look at the red-blue variants of this problem, where we wish to find an object with k red edges. This can be done for spanning trees in polynomial time – first find a tree with a maximal number of red edges (How?), find a tree with a minimal number of red edges, then perform swapping operations edge-by-edge to get from the first to the second, and at some point in this process there must be exactly k red edges if possible. This is NP-complete for paths, since it can be used to find a Hamiltonian path. Finally, we don't know whether this problem is in P or NP-complete for perfect matchings.
- Next, we have the counting variant of this problem. This can be done in polynomial time for spanning trees using the *matrix tree theorem*. This is #P-complete for boths paths and perfect matchings.
- Finally, let us look at the problem of determining the maximum number of edge-disjoint objects. This can again be done in polynomial time for spanning trees, in polynomial time for paths (using flow, say). For perfect matchings, this is in polynomial time for bipartite graphs but is NP-complete in general.

1.4. Forcing number

Definition 1.15 (Forcing number)**.** Given a perfect matching M in a graph G, the *forcing number* of M is the size of the smallest subset $S \subseteq M$ such that M is the only perfect matching containing S. In such a case, we say that S *forces* M.

This is related to the problem of setting a Sudoku, where given a solution, we are interested in the smallest number of clues required to make the solution unique. The forcing number is used in chemistry, where atoms are vertices and edges are bonds.
This is NP-complete for graphs in general, unfortunately. However, we can also study the *minimum forcing number* of a graph, which is the smallest forcing number across all perfect matchings of the graph, and we can similarly study the *maximum forcing number*.

Note that no alternating cycle of a matching can contain an edge in its forcing set. In particular, if a matching has k disjoint alternating cycles, its forcing number is at least k.

For the $2n \times 2n$ grid graph, it has been proved that the maximum forcing number is n^2 and the minimum forcing number is n. It was also proved that there exists a perfect matching with forcing number k for any integer $n \leq k \leq n^2$.
In this same paper, it was conjectured that in the n-hypercube, every perfect matching has forcing number 2^{n-2}. This was soon proved to be false, by showing matchings on the hypercube for which nearly all edges in the matching need to be fixed for a unique perfect matching (the conjecture says we only need to fix $1/2$ the edges). We still do not know the maximum forcing number of the hypercube, however.

We introduce a fairly general technique to prove lower bounds on the minimum forcing number of a bipartite graph. Consider the bipartite adjacency matrix A of the graph, and let M be a matching, and let $S \subseteq M$ be a forcing set of size k. Consider the $(n-k) \times (n-k)$ submatrix A' of A comprising those rows and columns (vertices) that are *not* in S. Then, the determinant of A' is ± 1. In fact, if we replace each 1 entry in A' with any nonzero value, the determinant remains nonzero.
That is, if there exists a matching with forcing number k, if we assign any nonzero weights to the edges, there always exists a $(n-k) \times (n-k)$ nonsingular submatrix, that is, the rank of A is at least $(n-k)$. In particular, the rank of the matrix is at least $(n-k)$ in any field, if we replace each 1 with a nonzero field entry. If we want to prove a lower bound on the minimum forcing number, we should assign nonzero weights to the edges in some manner that the rank of the resulting matrix is small.

Definition 1.16 (Min-rank)**.** Given a $n \times n$ $0,1$-matrix M, the *min-rank* of M is the smallest possible rank of M', where M' is obtained by replacing each nonzero entry of M with a nonzero field entry.

Therefore, n minus the min-rank of a graph's bipartite adjacency matrix is a lower bound on the graph's minimum forcing number.
So, a natural next question is: when is this method tight? For which graphs, with minimum forcing number k, is $(n-k)$ precisely the min-rank of the bipartite adjacency matrix?
Slightly more generally, for what bipartite graphs is the number of vertices in the largest subgraph with a unique perfect matching equal to twice the min-rank of the bipartite adjacency matrix? We do not know any examples where the two are not equal.

Definition 1.17 (Cartesian Product)**.** Given graphs G_1, G_2, the *Cartesian product* $G_1 \times G_2$ of the graphs has vertex set $V(G_1) \times V(G_2)$, with (u_1, v_1) adjacent to (u_2, v_2) iff $u_1 = u_2$ and $v_1 \leftrightarrow v_2$ or $v_1 = v_2$ and $u_1 \leftrightarrow u_2$.

That is, we replace each vertex in G_2 by a copy of G_1, also making the copies adjacent for neighbouring vertices in G_2. The graph $G \times K_2$ is sometimes called the *prism* over G.
Given this, $Q_n = K_2 \times K_2 \times \cdots \times K_2$.

Lemma 1.18. Let G be a bipartite graph with the $n \times n$ bipartite adjacency matrix A. Then, the prism over G has bipartite adjacency matrix

$$\begin{pmatrix} A & \mathrm{Id}_n \\ \mathrm{Id}_n & A^\top \end{pmatrix}.$$

The proof of the above is straightforward and we omit it.

Denote by A_n the bipartite adjacency matrix of the n-hypercube. It is easily argued using the above lemma that A_n is symmetric.

Lemma 1.19. Over $\mathbb{F}_2$,

$$A_n^2 = \begin{cases} 0, & n \text{ is even,} \\ \mathrm{Id}, & n \text{ is odd.} \end{cases}$$

Proof. We prove this via induction. The base case when $n = 1$ is trivial. In general,

$$A_n^2 = \begin{pmatrix} A_n & \mathrm{Id} \\ \mathrm{Id} & A_n \end{pmatrix}^2 = \begin{pmatrix} A_n^2 + \mathrm{Id} & 2A_n \\ 2A_n & A_n^2 + \mathrm{Id} \end{pmatrix}.$$

Over $\mathbb{F}_2$, this is just $\mathrm{Id} - A_n^2$, so the required follows by the inductive hypothesis. ■

In particular, when n is odd, A_n has full rank over $\mathbb{F}_2$.

Lemma 1.20. The minimum forcing number of the n-hypercube Q_n is at least 2^{n-1}.

Proof. Suppose n is even. We shall prove that the rank of A_n over $\mathbb{F}_2$ is 2^{n-1}. Recalling that $A_{n-1}^2 = \mathrm{Id}$,

$$\begin{pmatrix} A_{n-1} & -\mathrm{Id} \\ 0 & \mathrm{Id} \end{pmatrix} \begin{pmatrix} A_{n-1} & \mathrm{Id} \\ \mathrm{Id} & A_{n-1} \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ \mathrm{Id} & A_{n-1} \end{pmatrix}.$$

Since the first matrix is non-singular due to the non-singularity of A_{n-1}, the rank of A_n is half the number of rows of A_n, that is, 2^{n-1}.

When n is odd, this argument does not work directly. Here however, we leverage the fact that we can assign nonzero weights to the nonzero entries. Replacing the entries of the A_{n-1} parts, we shall get

$$\begin{pmatrix} W_{n-1} & \mathrm{Id} \\ \mathrm{Id} & W_{n-1}^{-1} \end{pmatrix},$$

so that we can apply the same trick as in the first part to show that the rank of this matrix is half the number of rows of A_n. That is, we would like to be able to assign nonzero weights to the nonzero entries of W_{n-1} such that the inverse of W_{n-1} is also obtained by assigning nonzero weights to the nonzero entries. More succinctly, the "nonzero pattern" of W_{n-1}, W_{n-1}^{-1}, and A_{n-1} are identical.
This can inductively be done rather easily however. We have $W_1 = W_1^{-1} = \mathrm{Id}$. In general, we define

$$W_n = \begin{pmatrix} 2W_{n-1} & \mathrm{Id} \\ \mathrm{Id} & W_{n-1}^{-1} \end{pmatrix},$$

so

$$W_n^{-1} = \begin{pmatrix} W_{n-1}^{-1} & -\mathrm{Id} \\ -\mathrm{Id} & 2W_{n-1} \end{pmatrix}$$

has the same nonzero pattern as A_n by the inductive hypothesis. ■

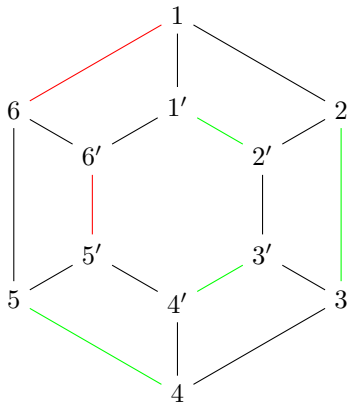
Note that we can take the W_n modulo p for any prime $p \neq 2$.

Let us next consider a $n \times n$ bipartite graph G. It is easy to see that the minimum forcing number of $G \times K_2$ is at most n, by considering the matching which matches a vertex to its copy – if we map all the vertices on one side of the bipartition to its copy, the remaining n edges in the matching are forced by the fact that there are no edges remaining.

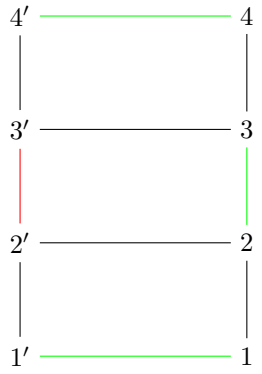
Question 1. For which bipartite graphs does the corresponding prism have minimum forcing number exactly n? For such a graph, can this be proved using the min-rank argument?

The second part amounts to finding a $n \times n$ weight matrix W such that W has the same nonzero pattern as A and W^{-1} has the same nonzero pattern as $A^\top$.

For the first question, there do exist graphs whose prisms don't have minimum forcing number exactly n.



(a) The minimum forcing number of $C_6 \times K_2$ is less than 3



(b) The minimum forcing number of $P_4 \times K_2$ is 1.

Figure 1: The coloured edges form a matching with the red edges being a forcing set.

If the forcing number of a matching is k, there exists an induced subgraph on $2(n-k)$ vertices with a unique perfect matching. The converse of this need not be true, since we also require that the subgraph on the remaining $2k$ vertices has a perfect matching. For example, the $2 \times n$ grid graph has an induced subgraph on $2n-2$ vertices with a unique perfect matching.

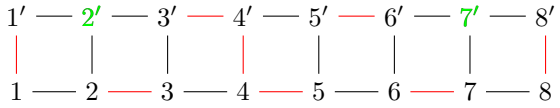


Figure 2: The subgraph on $2n-2$ vertices has a unique perfect matching but the minimum forcing number is > 1.

A *conformal subgraph* is a subgraph such that the subgraph obtained by removing its vertices has a perfect matching. Therefore, if the forcing number of a matching is k, there exists an induced conformal subgraph on $2(n-k)$ vertices with a unique perfect matching.

Question 2. Then, which graphs are such that if their minimum forcing number is k, there exists a largest induced subgraph with a unique perfect matching of size $2k$? That is, when does there exist a largest induced subgraph with a unique perfect matching that is also conformal?

Question 3. When is the order of the largest induced subgraph with a unique perfect matching equal to twice the min-rank of the bipartite adjacency matrix?

Let $G(A, B)$ be a bipartite graph over $2n$ vertices. For a prism $G \times K_2$, the minimum forcing number is at most n, for example by matching corresponding vertices in A between the two copies of G. If we find a matrix W that has the same non-zero pattern as A such that W^{-1} has the same nonzero pattern as $A^\top$, then the largest induced subgraph with a unique perfect matching has order $2n$, which implies that the minimum forcing number is n. A necessary condition for such a W to exist is that no two rows of A have exactly 1 column in which both rows are 1. That is, no two vertices in G have exactly one common neighbour.

Question 4. Is the above necessary condition sufficient? That is, if we have a $0,1$-matrix A such that no two rows have exactly one coordinate where both are 1, then does there exist a matrix W with the same nonzero pattern as A such that W^{-1} has the same nonzero pattern as $A^\top$? If yes, can we ensure that $W^{-1} = \lambda W^\top$ for some scalar λ?

For example, in the complete bipartite graph $K_{n,n}$ which has bipartite adjacency matrix equal to J (the all 1s matrix), an example of such a matrix W (for $n = 3$, say) is just

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & \omega & \omega^2 \\ 1 & \omega^2 & \omega \end{pmatrix},$$

where $\omega = e^{2\pi\iota/3}$.
Another example is $K_{n,n}$ with a perfect matching removed (for $n > 3$). For $n = 4$, over $\mathbb{F}_2$, the resulting bipartite adjacency matrix is its own inverse. In general, we can take the matrix itself over $\mathbb{F}_p$ for p any prime dividing $n - 2$.

What happens if we take any bipartite graph G', and then modify it to a graph G by duplicating all the vertices and adding edges uv' if uv is an edge, where v' is the copy of v? The bipartite adjacency matrix of this graph is symmetric on taking an appropriate ordering of the vertices.

This question arises from the study of *Hadamard matrices*, which are $n \times n$ ± 1-matrices H such that $HH^\top = n\,\mathrm{Id}$. There is a famous conjecture stating that such an H exists for $n \equiv 0 \pmod 4$. There is an easy construction for $n = 2^k$, by taking $H_1 = \begin{pmatrix} 1 \end{pmatrix}$ and

$$H_{2^k} = \begin{pmatrix} H_{2^{k-1}} & H_{2^{k-1}} \\ H_{2^{k-1}} & -H_{2^{k-1}} \end{pmatrix}.$$

An even more general notion is that of *weighing matrices* W, where we allow the entries to be in $\{0, \pm 1\}$ instead of just ± 1 like in Hadamard matrices, and we are interested in having $WW^\top = w\,\mathrm{Id}$ for some integer w. That is, any row has exactly w nonzero entries, and any two rows are orthogonal to each other.

§2. Combinatorial Optimization

"Optimization" is the subject where we attempt to maximize/minimize some function. "Combinatorial" refers to the fact that the function we are trying to optimize over a finite set.
In matching theory, there is a plethora of "min-max" results, where the maximum of some function is associated to the minimum of another.

Definition 2.1. Given a bipartite graph $G(A, B)$, define the *deficiency* of G by

$$\mathrm{def}(G) = \max_{X \subseteq A} |X| - |N(X)|.$$

Note that $\mathrm{def}(G) \geq 0$ for any graph, since $|\varnothing| - |N(\varnothing)| = 0$.

Proposition 2.2. The size of the largest matching in a bipartite graph $G(A, B)$ is equal to $|A| - \mathrm{def}(G)$.

Arguably the first min-max result in matching theory was the following.

Theorem 2.3 (König's Theorem)**.** The size of a maximum matching in a bipartite graph is equal to the minimum number of vertices required to cover all edges in the graph (a minimum vertex cover) – the smallest subset such that every edge has at least one endpoint in the subse.t

This result is equivalent to Hall's Theorem.
Proofs of min-max results usually go about by showing that one quantity is always larger than the other, then proving that there is some object of each type whose corresponding quantities are equal.
Consider the problem of finding a maximum weight perfect matching – given positive weights (w_e) assigned to edges, find a matching of maximum weight. This is associated to a minimum problem using LP duality.
For each edge, associate a variable $x_e \in \{0, 1\}$ indicating whether an edge is in the matching or not. Then, finding a max weight matching amounts to the program

$$\begin{aligned} \text{maximize} \quad & \sum_{e \in E} w_e x_e \\ \text{subject to} \quad & \sum_{e \text{ incident on } v} x_e \leq 1, \qquad v \in V \\ & x_e \in \{0, 1\}, \qquad e \in E \end{aligned}$$

This is an *integer linear program*, which is NP-complete in general. We can get a normal linear program by relaxing the integrality constraint, say by allowing the x_e variables to take any non-negative value.

$$\begin{aligned} \text{maximize} \quad & \sum_{e \in E} w_e x_e \\ \text{subject to} \quad & \sum_{e \text{ incident on } v} x_e \leq 1, \qquad v \in V \\ & x_e \geq 0, \qquad e \in E \end{aligned} \tag{2.1}$$

The *dual* of the above linear program (2) is formed by taking a variable $y_v \geq 0$ for each vertex v, and multiplying the inequality corresponding to v with y_v. This changes the second constraint to

$$\sum_{e \text{ incident on } v} y_v x_e \leq y_v.$$

Adding this up over all vertices gives

$$\sum_{\text{edges } e=uv} (y_u + y_v) x_e \leq \sum_{v \in V} y_v.$$

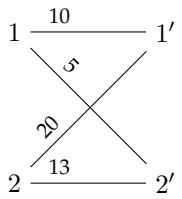
If the y_v are such that for any edge uv, $y_u + y_v \geq w_e$, then

$$\sum_{e=uv} w_e x_e \leq \sum_{e=uv} (y_u + y_v) x_e \leq \sum_v y_v.$$

From the perspective of König's Theorem where all the w_e are 1, this just asks for a smallest vertex cover!
We get the dual linear program

$$\begin{array}{lll} \text{minimize} & \sum_{v \in V} y_v & \\ \text{subject to} & y_u + y_v \geq w_e, & uv = e \in E \\ & y_v \geq 0, & v \in V \end{array} \tag{2.2}$$

For example, consider the following graph.



The linear program (2) is optimized by setting $(x_{11'}, x_{12'}, x_{21'}, x_{22'}) = (0, 1, 1, 0)$, that is, the matching $\{12', 21'\}$. On the other hand, the dual linear program (2) is optimized by setting $(y_1, y_2, y_1', y_2') = (4, 13, 7, 1)$. Note that in either case, the optimum is equal to 25!

More generally, we have the following definition.

Definition 2.4. Given the *primal* linear program

$$\begin{array}{ll} \text{maximize} & c^\top x \\ \text{subject to} & Ax \leq b, \\ & x \geq 0, \end{array}$$

its *dual* is

$$\begin{array}{ll} \text{minimize} & b^\top y \\ \text{subject to} & A^\top y \geq c, \\ & y \geq 0. \end{array}$$

Note that the dual of the dual of a primal is the primal itself.

Proposition 2.5 (Weak Duality)**.** For any feasible solution x of the primal and y of the dual, $c^\top x \leq b^\top y$.

Section 2 can more succinctly be written as

$$\begin{array}{ll} \text{maximize} & w^\top x \\ \text{subject to} & Ax \leq 1, \\ & x \geq 0, \end{array}$$

where A is the $n \times m$ incidence matrix of the graph, with $A_{ij} = 1$ iff edge e_j is incident on v_i. The dual LP (corresponding to section 2) is

$$\begin{array}{ll} \text{minimize} & 1^\top y \\ \text{subject to} & A^\top y \geq w, \\ & y \geq 0. \end{array}$$

Given a linear program (as in Definition 2.4, say), the *feasible region* of the LP is $\{x \in \mathbb{R}^m : Ax \leq 1, x \geq 0\}$, which is a convex polytope – the convex hull of finitely many points. A *vertex* of a polytope is a point in it that is not a convex combination of two distinct points in the polytope. It may be shown that any linear program has an optimum at a vertex of the polytope. More generally, the set of points where the optimum is attained is some face of the polytope. An *integral polytope* is one whose vertices have integral coordinates.

Definition 2.6. Given a graph G, the polytope defined by section 2 is integral iff G is bipartite.

Proof. For the forward direction, we have that if the graph has an odd cycle, the point that assigns $1/2$ to the edges in the cycle and 0 to the remaining vertices is a vertex of the polytope. Indeed, if it were a convex combination of two distinct points in the polytope, both points are forced to have the coordinates of all edges not in the cycle as 0, and if such a point in the polytope assigned $1/2 + \epsilon$ to some edge, then the next edge in the cycle can be assigned at most $1/2 - \epsilon$. Repeating this argument around the cycle yields that the other edge adjacent to the edge with weight $1/2 + \epsilon$ has weight at least $1/2 + \epsilon$, a contradiction. ■

We next give an optimality criterion for a linear program.

Lemma 2.7 (Complementary slackness)**.** Suppose that x, y are solutions to a primal LP and its dual such that

(a) if $y_v > 0$, the vth inequality in the primal is tight for x and

(b) if $x_e > 0$, the eth inequality in the dual is tight for y.

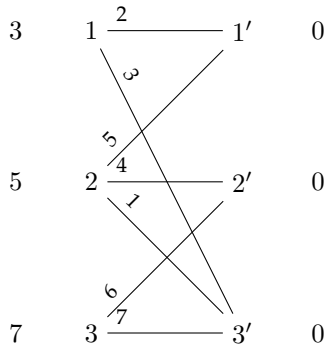
In this case, the value of the primal for x equals the value of the dual for y, so both have the same optimum.

In the context of matchings, this means that if some edge e has positive x_e, then the sum of y values of its endpoints is equal to the weight w_e. Similarly, if a vertex has positive y_v, the sum of x_e values over edges e incident on v is equal to 1.

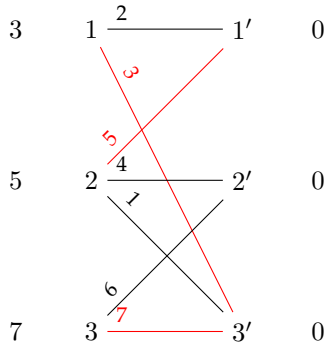
Proof of Lemma 2.7 in the setting of matchings. The proof is immediate since

$$\sum_e w_e x_e = \sum_{\substack{e=uv \\ x_e > 0}} w_e x_e = \sum_{\substack{e=uv \\ x_e > 0}} (y_u + y_v) x_e = \sum_u y_u \sum_{e \text{ incident on } u} x_e = \sum_{u : y_u > 0} y_u. \quad ■$$

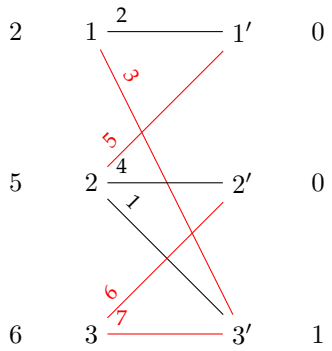
Let us now give an algorithm for finding a max-weight matching in the bipartite setting. Start off with the dual solution y, where y_u is the maximum weight edge incident on u if $u \in A$ and 0 otherwise.



We will try to maintain complementary slackness by allowing only those edges take nonzero value. Consider the set of *admissible* edges whose A vertex have them as the maximum weight edge. Find a maximum cardinality matching in the graph formed by admissible edges – this can be done by any old algorithm such as that which searches for augmenting paths.



If this is is a matching from A to B, it is a maximum weight matching. If not, there exists a subset X such that $|N(X)| < |X|$. Then, increase the y value of vertices in $N(X)$ by δ and decrease the y value of vertices in X by δ, for the largest possible δ. This means that we increase/decrease it until the weight of some vertex becomes 0, or some new edge becomes admissible. In the former case, we delete that vertex from the graph.



Now, how large can the δ mentioned earlier get? First off, none of the y values can become negative, so define

$$\delta_1 = \min_{v \in A' \cup \{u\}} y_v.$$

We also have the constraint due to formerly non-admissible edges pq that might become tight due to the decrease in y_p for some $p \in A'$. So, define

$$\delta_2 = \min_{\substack{pq \text{ not admissible} \\ p \in A', q \notin B'}} (y_p + y_q - w_{pq}).$$

Then, we set $\delta = \min\{\delta_1, \delta_2\}$, decreasing y_u by δ for $u \in A'$, and increasing y_v by δ for $v \in B'$. We repeat this process until all vertices with positive y value are matched.

Now, in the problem of finding a maximum weight *perfect* matching, we instead have

$$\begin{aligned} \text{maximize} \quad & \sum_{e \in E} w_e x_e \\ \text{subject to} \quad & \sum_{e \text{ incident on } v} x_e = 1, \qquad v \in V \\ & x_e \geq 0, \qquad e \in E \end{aligned}$$

and in the dual,

$$\begin{aligned} \text{minimize} \quad & \sum_{v \in V} y_v \\ \text{subject to} \quad & y_u + y_v \geq w_e, \qquad uv = e \in E. \end{aligned}$$

Note that the $y_v \geq 0$ constraint has been removed. The algorithm now is exactly the same as the earlier one for a maximum weight matching, except that the y coordinates are allowed to be negative.

This is an $O(mn)$ time algorithm, called the *Hungarian method*. Some mild optimizations can bring it down to $O(m\sqrt{n})$. Recently, we got an almost linear time algorithm to find a maximum weight perfect matching!
Now, the problem of finding maximum weight matchings can be reduced to finding maximum weight perfect matchings by adding extra vertices with the edges having 0 weight.

In general graphs, we have

$$\begin{aligned} \text{maximize} \quad & w^\top x \\ \text{subject to} \quad & \sum_{e \text{ incident on } v} x_e = 1, \qquad v \in V \\ & x_e \geq 0, \qquad e \in E \end{aligned}$$

This is in general not an integral polytope. If the graph has no odd cycles, it is bipartite so integral, by an argument similar to the previous one. Even in the case where it does have odd cycles, the graph could be integral, an example being K_4.
Generally, graphs for which this polytope is integral are called *Birkhoff-von Neumann* graphs. There is no characterization known for such graphs.
Suppose we have a graph with two odd cycles, with the remaining vertices having a perfect matching among themselves. Then, the point assigning 1 to the matching and $1/2$ to all vertices in the odd cycle is a vertex of the polytope.

Lemma 2.8. Let G be a matching-covered graph, that is, a connected graph where every edge is in a perfect matching. The polytope defined by (2) is not integral iff G contains evenly many disjoint odd cycles such that the remaining graph has a perfect matching.

This gives some sort of co-NP characterization of which graphs have integral perfect matching polytopes, but it is unclear how to check this property.

Definition 2.9. Given a graph $G = (V, E)$ with a perfect matching, identify each perfect matching M of G with the vector $x \in \mathbb{R}^E$, where $x_e = 1$ if $e \in M$ and 0 otherwise. The *perfect matching polytope* of G is the convex hull of these points. The *matching polytope* is defined similarly.

However, this polytope may have exponentially many vertices, so we need some more succinct way to describe it. It turns out that the polytope has far fewer faces than vertices, so it can easily be described by a collection of linear (in)equalities. Indeed, we just need to add appropriate inequalities to (2), using Lemma 2.8.

Theorem 2.10. Given a graph G, its perfect matching polytope is the set x of points in $\mathbb{R}^E$ satisfying

$$x_e \geq 0 \qquad \text{for all } e \in E, \tag{2.3}$$

$$\sum_{e \text{ incident on } v} x_e = 1 \qquad \text{for all } v \in V, \tag{2.4}$$

$$\sum_{e \in \partial S} x_e \geq 1 \qquad \text{for all odd sized } S \subseteq V \text{ with } 1 < |S| \leq n/2. \tag{2.5}$$

Show that the third inequality discounts vertices in Lemma 2.8. In fact, the above constraint only needs to be added for certain S, not necessarily all S. However, which S we must add depends on the graph itself. Indeed, when we take the dual, most variables are immediately set to 0 so not many variables need to be considered. For the complete graph K_n on evenly many vertices for instance, we do need to add every such constraint.
Although this is exponentially many constraints, we can still sometimes efficiently solve a linear program. For example, the problem of finding a minimum weight spanning tree can be solved efficiently, although the *spanning tree polytope*

$$\sum_{e \in G[S]} x_e \leq (|S| - 1) \qquad \text{for all } \varnothing \neq S \subseteq V,$$

$$\sum_{e \in E} x_e = (n - 1).$$

has exponentially many constraints. Above, $G[S]$ denotes the subgraph of G induced by S. Alternatively, we could replace the first constraint with the constraint that for any partition $\mathcal{P}$ of the vertex set, the sum of x_e over edges crossing the partition is at least $|\mathcal{P}| - 1$. Kruskal's and Prim's algorithm to find a minimum weight spanning tree can in fact be expressed as appropriate linear programs.